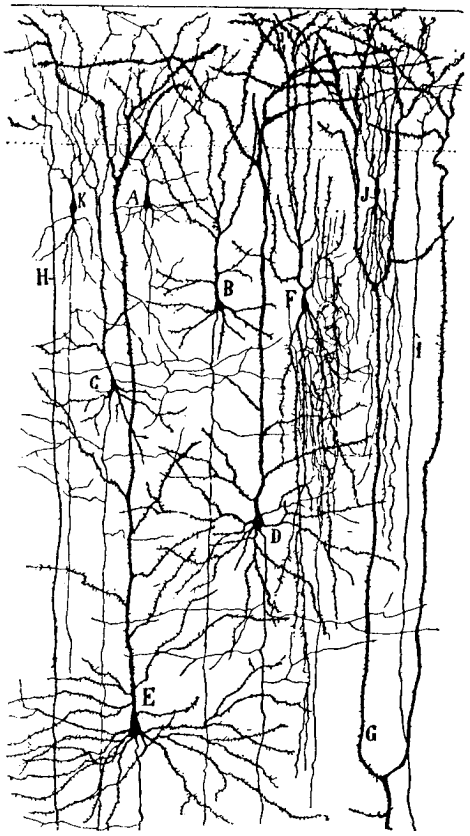
Neural Networks

Lecture Notes of Course CM451Z

October 1998



A.C.C. Coolen
Department of Mathematics
King's College London

Contents

With contributions by
H.C. Rae, N. Skantzos and A. Heimel

London, March 21, 2001

Chapter 1

General Introduction

1.1 Neural Information Processing

The brain is a piece of hardware that performs sophisticated information processing tasks, using microscopic elements and operations which are fundamentally different from the ones on which present-day computers are based. The microscopic processors in the brain, the brain cells, or *neurons* (see figure 1.1) are rather noisy elements which operate in parallel. They do not execute a fixed 'program' on a given set of 'data', but communicate signals trough relay stations (the *synapses*, or *synaptic efficacies*), located at the junctions where the output channel (*axon*) of one neuron meets an input channel (*dendrite*) or the cell body of another. The strengths of the relay stations are continuously being updated, albeit slowly. The neurons of each given brain region are organised and wired in a specific network, the structure of which can vary from very regular (especially in regions responsible for pre-processing of sensory data) to almost amorphous (especially in the 'higher' regions of the brain, where cognitive functions are performed), see figure 1.2. These dynamic relay stations, or synapses, in combination with some adjustable intrinsic neuron properties, represent both 'data' and 'program' of the network; consequently, program and data change all the time.

We can roughly summarise similarities and differences between conventional present-day computer systems and biological neural networks in the following table:

computers	biological neural networks
processors *operation speed* $\sim 10^8 Hz$ *signal/noise* $\sim \infty$ *signal velocity* $\sim 10^8 m/sec$ *connections* ~ 10	neurons *operation speed* $\sim 10^2 Hz$ *signal/noise* ~ 1 *signal velocity* $\sim 1 m/sec$ *connections* $\sim 10^4$
sequential operation program & data external programming	parallel operation connections and neuron characteristics self-programming & adaptation
not robust against hardware failure cannot deal with unforseen data	robust against hardware failure messy, unforseen and inconsistent data

From an engineering point of view the neurons are clearly extremely poor substitutes for

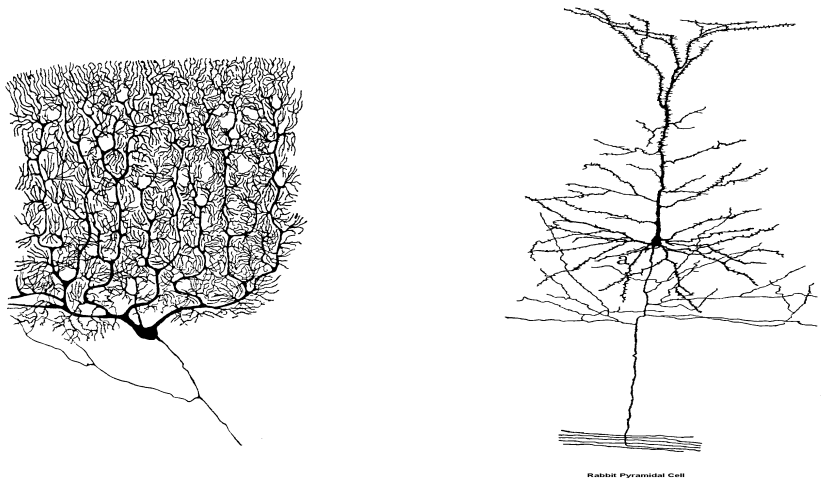


Figure 1.1: Left: a Purkinje neuron in the human cerebellum (a brain region responsible for smooth movement control); Right: a Pyramidal neuron in the rabbit cortex (the region of cognitive functions). Both pictures are due to Ramon y Cajal (1880), obtained from a chemical staining method, invented by Golgi. The black blobs are the neuron cell bodies, the trees of wires fanning out constitute the input channels (or 'dendrites') through which signals are received, sent off by other firing neurons. The lines at the bottom of the pictures, bifurcating only modestly, are the output channels (or 'axons').

processors; they are several orders of magnitude more slow and unreliable. In the brain this setback is overcome by redundancy: by making sure that always a very *large* number of neurons are involved in any process, and by preferably having them operate in *parallel*. This is in contrast to conventional computers, where individual operations are as a rule performed sequentially, i.e. one after the other, so that failure of any part of this chain of operations is mostly fatal. The other fundamental difference is that conventional computers can only execute a detailed specification of orders, the program, requiring the programmer to know exactly which data can be expected and how to respond. Any subsequent change in the actual situation, not foreseen by the programmer, leads to trouble. Neural networks, as we know from everyday experience, can adapt quite well to changing circumstances. We can recognise objects also if they are deformed or only partly visible, our visual system can adapt to the strangest deformations of the image on our retina (for instance: everything upside down), we can even rewire the nerve fibres coming from arms and legs and again learn how to control movements. Finally, there is the robustness against physical hardware failure. In our brain large numbers of neurons end their careers each day unnoticed. Compare this to what happens if we randomly cut a few wires in our workstation.

We know what the brain and its neural network building blocks can do, the question now is: how does it do these things ? It has been suggested already in 1957 by von Neumann that, in view of the large number of interacting neurons, (in the order of 10^{11}, each of which communicating with roughly 10^4 colleagues) and the stochastic nature of neural processes, statistical theories might be the appropriate language for describing the operation of the

Figure 1.2: Pyramidal neurons in the visual area of the cortex (from Ramon y Cajal (1880)). Golgi's staining method colours only a fraction of the neurons, so in reality the network is more dense than the picture suggests.

brain [1]. Later such ideas were given a more precise meaning in terms of stochastic processes and statistical mechanics. Roughly speaking, one can distinguish three types of motivation for studying neural networks. Biologists aim at understanding information processing in real biological nervous tissue. Engineers and computer scientists would like to use the principles behind neural information processing for designing adaptive software and artificial information processing systems which can learn and of which the processors can operate efficiently in parallel. Such devices would clearly be quite complementary to the conventional types of computers. Theoretical physicists and mathematicians are challenged by the fundamental new problems posed by neural network models, which exhibit a highly non-trivial and rich behaviour. Consequently, the types of model studied by the different groups of scientists and the language in which they are formulated, as well as the role of experiments in guiding and constraining research will be different. Assumptions and approximations which are fruitful and natural to the biologist can be useless or even forbidden in the context of artificial systems, and vice versa. Neural networks are rather complex systems to analyse for several reasons: (*i*) the large number of interacting elements, (*ii*) the non-linear character of the operation of the individual elements, (*iii*) the interactions between the elements are not identical, or at least periodic in space, but usually different in strength for each individual pair of elements, (*iv*) two given neurons can operate on one another in a different way (there is not even pairwise symmetry), and (*v*) the interactions and firing thresholds change all the time. There are two main strategies to simplify analysis. The first is to look at *layered networks*, where no interaction loops are present, so that the states of the neurons can be

[1] The numbers quoted here are not to be taken too literally, their exponents vary in between textbooks.

calculated iteratively, layer by layer. The second is to describe the system statistically at a *macroscopic* level, of global quantities, and forget about the microscopic details at the level of the behaviour of individual neurons.

Biological Modelling. Here there is still a huge gap between theory and experiment. It is not at all clear which are the full microscopic laws. Of those microscopic processes that have been identified, mostly relating to the operation of individual neurons, we do not know which degrees of freedom are relevant and which are 'accidental' (i.e. non-productive artifacts of evolution). Last but not least, one often does not know how to quantify the macroscopic processes, i.e. what to look for. In order to arrive at a level of description where mathematical analysis becomes possible, specific choices of simplifications have been made in model studies of neural networks. In the typical model, neuron states are represented by scalar variables[2], which evolve in time stochastically, driven by so-called post-synaptic potentials, which are usually taken to depend linearly on the states of the neurons. In recurrent networks there is no simple feedforward-only or even layered operation, but rather the neurons drive one another collectively and repetitively without particular directionality. In these networks the interest is in the global behaviour of all the neurons and the associative retrieval of memorised states from initialisations in noisy representations thereof. They are often referred to as *attractor* neural networks[3]. They are idealisations of parts of the brain, such as cerebral cortex. The study of this type of model has resulted in a thorough (even quantitative) understanding of especially the functioning of nervous tissue as associative memories for static and dynamic patterns, but also in insight into typically biological phenomena like memory disorders induced by damage, phase-locked neural oscillations and chemical modulation. The simplifications introduced to arrive at solvable models are more or less ad hoc, motivated by experience, intuition, and the desire to quantify the problem. However, the last decade has shown that progress is indeed being made in incorporating more biological detail into analytically solvable models.

Artificial Neural Networks. Here one is not particularly interested in the chemical and electrical details of neural information processing, but rather in understanding the two building blocks of learning and massive parallellism, on which the remarkable computational power of the brain is based. Biological experiments are only relevant in that they might hint at possible mechanisms for achieving the aim, but play no role as constraints on model ingredients. Rather than a necessary simplification, to the computer scientists representing neuron states as binary variables is a welcome translation into familiar language. The preferred architecture of many artificial neural networks for application as expert systems is that of layers. Here many input neurons drive various numbers of hidden units eventually to one or few output neurons, with signals progressing only forward from layer to layer, never backwards or sideways within a layer. The interest lies in training and operating the networks for the deduction of appropriate few-state conclusions from the simultaneous input of many, possibly corrupted, pieces of data.

[2]Especially among psychologists, the scalar information processing units in idealised neural network models are assumed not to represent individual neurons, but rather groups of neurons.

[3]They are often abbreviated as ANN, but we avoid this notation since it is also common for *artificial* neural networks.

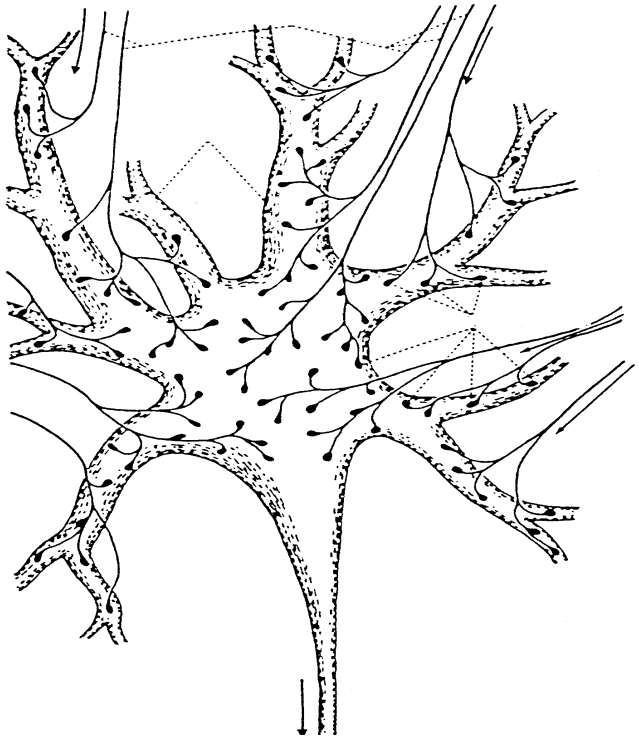


Figure 1.3: Schematic drawing of a simple neuron, with its dendritic tree (branching from the cell body), the incoming axons from other neurons (incoming solid lines) with their points of contact (synapses; dark blobs), and its own axon (outgoing branch at the bottom).

1.2 Biological Neurons and Model Neurons

Some Biology. Neurons come in all kinds of shapes and sizes, but, roughly speaking, they all operate more or less in the following way (see figure 1.3). The lipidic cell membrane of a neuron maintains concentration differences between inside and outside the cell, of various ions (the main ones are Na^+, K^+ and Cl^-), by a combination of the action of active ion pumps and controllable ion channels. When the neuron is at rest, the channels are closed, and due to the activity of the pumps and the resultant concentration differences, the inside of the neuron has a net negative electric potential of around –70 mV, compared to the fluid outside.

equilibrium concentrations (mmol/lt)	Na^+	K^+	Cl^-
outside the cell	143	5	103
inside the cell	24	133	7

A sufficiently strong local electric excitation, making the cell potential temporarily less negative, leads to the opening of specific ion channels, which in turn causes a chain reaction of other channels opening and/or closing, with as a net result the generation of an electrical peak of height +40 mV, with a duration of about 1 msec, which will propagate along the membrane at a speed of about 5 m/sec: the so-called *action potential.* After this electro-chemical

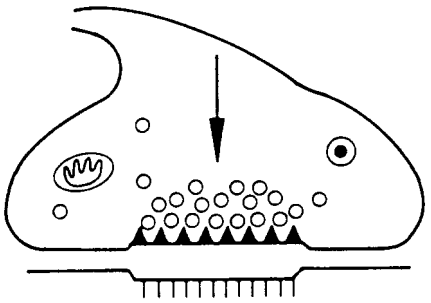


Figure 1.4: Schematic drawing of a simple synapse. Upper part: terminal of the axon of the sending neuron, bottom: surface of a dendrite of the receiving neuron (the two are separated by the so-called synaptic cleft). Circles: pockets of neurotransmitter, to be released in synaptic cleft upon the arrival of an action potential along the axon.

avalanche it takes a few milliseconds to restore peace and order, during this period, the so-called *refractory period*, the membrane can only be forced to generate an action potential by extremely strong excitation. The action potential serves as an electrical communication signal, propagating and bifurcating along the output channel of the neuron, the *axon*, to other neurons. Since the propagation of an action potential along an axon is the result of an active electro/chemical process, the signal will retain shape and strength, even after bifurcation, much like a chain of tumbling domino stones.

The junction between an output channel (axon) of one neuron and an input channel (dendrite) of another neuron, is called *synapse* (see figure 1.4). The arrival of an action potential can trigger the release of a chemical, the *neurotransmitter*, into the so-called *synaptic cleft* which separates the cell membranes of the two neurons. The neurotransmitter in turn acts to open selectively ion channels in the membrane of the dendrite of the receiving neuron. If these happen to be Na^+ channels, the result is a local increase of the potential at the receiving end of the synapse, if these are Cl^- or K^+ channels the result is a decrease. In the first case the arriving signal will increase the probability of the receiving neuron to start firing itself, therefore such a synapse is called *excitatory*. In the second case the arriving signal will decrease the probability of the receiving neuron being triggered, and the synapse is called *inhibitory*. The main neurotransmitters are now believed to be glutamate (operating in excitatory synapses) as well as GABA (gamma-amino butyric acid) and clycine (operating in inhibitory synapses). However, there is also the possibility that the arriving action potential will not succeed in releasing neurotransmitter; neurons are not perfect. This introduces an element of uncertainty, or noise, into the operation of the machinery. A general rule (Dale's Law) is that every neuron can have only one type of synapse attached to the branches of its axon; it either excites all neurons it sends signals to (in which case it is called an *excitatory neuron*), or it inhibits all neurons it sends signals to (in which case is is called an *inhibitory neuron*).

Whether or not the receiving neuron will actually be triggered, will depend on cumulative effect of all excitatory and inhibitory signals arriving, a detailed analysis of which requires also taking into account the electrical details of the dendrite. The region of the neuron membrane most sensitive to be triggered into sending an action potential is the so-called

hillock zone, near the root of the axon. If the potential in this region, the *post-synaptic potential* exceeds some neuron-specific threshold (of the order of –30 mV), the neuron will fire an action potential. However, the firing threshold is not a strict constant, but can vary randomly around some average value (so that there will always be some non-zero probability of a neuron not doing what we would expect it to do with a given postsynaptic potential), which constitutes the second main source of uncertainty into the operation.

The key to the adaptive and self-programming properties of neural tissue and to being able to store information, is that the synapses and firing thresholds are not fixed, but are being updated all the time. It is not entirely clear how this is realised at a chemical/electrical level. Most likely the amount of neurotransmitter in a synapse, available for release, and the effective contact surface of a synapse are modified.

Let us conclude with some numbers to get an idea of dimensions:

characteristic time-scales		typical sizes	
duration of action potential:	$\sim 1msec$	neuron cell body:	$\sim 50\mu m$
refractory period:	$\sim 3msec$	axon diameter:	$\sim 1\mu m$
synaptic signal transmission:	$\sim 1msec$	synapse size:	$\sim 1\mu m$
axonal signal transport:	$\sim 5m/sec$	synaptic cleft:	$\sim 0.05\mu m$

Model Neurons. Although we can never be sure beforehand to which level of microscopic detail we will have do descend in order to understand the emergent global properties of neural networks, there are reasons for not trying to analyse in all detail all chemical and electrical processes involved in the operation of a given neural system. Firstly, we would just end up with a huge set of nasty equations that are impossible to handle, and consequently learn very little. Secondly, the experiments involved are so complicated that the details we would wish to translate into equations are still being updated frequently.

Let us now first try to construct a simple neuron model. Not all of the simplifications we will make along the way are strictly necessary, and some can be removed later. Note that what follows is certainly not a strict derivation, but rather a rough sketch of how various neuron models can be related to biological reality. Our neuron is assumed to be embedded in a network of N neurons, which will be labelled with the index $i = 1, \ldots, N$. The post-synaptic potential of our neuron at time t will be called $V(t)$.

- First of all, we forget about the details of the avalanche creating an action potential, and concentrate only on the presence/absence of an action potential, denoted by the variable $S \in \{0, 1\}$:

$$\begin{array}{ll} S(t) = 1: & \text{neuron fires an action potential at time } t \\ S(t) = 0: & \text{neuron is at rest at time } t \end{array} \tag{1.1}$$

  If we denote the firing threshold potential of our neuron (which can vary) by $V^*(t)$, we can relate the firing state $S(t)$ to the post-synaptic potential as

$$S(t) = \theta[V(t) - V^*(t)] \tag{1.2}$$

  with the step function: $\theta[x] = 1$ for $x > 0$, $\theta[x] = 0$ for $x < 0$ (we can define $\theta[0]$ to be either 0, $\frac{1}{2}$ or 1).

- The second simplification we make is to forget about the possibility of any sender having multiple synaptic contacts with any receiver, and neglect the microscopic electrochemical details of the conversion at the synapses of arriving action potential into electric currents. This allows us to represent the synaptic interaction between our model neuron and any neuron k by a single real number $J_k \in \langle -\infty, \infty \rangle$:

$$\begin{array}{ll} J_k > 0: & \text{synapse connecting to the axon of } k \text{ is excitatory} \\ J_k < 0: & \text{synapse connecting to the axon of } k \text{ is inhibitory} \\ |J_k|: & \text{proportional to the magnitude of the resulting electric current} \end{array} \tag{1.3}$$

Consequently, $J_k = 0$ represents the case where a synapse is simply absent. Taking into account the possibility of an arriving action potential failing, the actual electric current $I_k(t) \in \langle -\infty, \infty \rangle$ injected into our model neuron, by neuron k at time t, can be written as:

$$I_k(t) = p_k(t) J_k S_k(t - \tau_k) \tag{1.4}$$

with

$$\begin{array}{ll} p_k(t) \in \{0, 1\}: & \text{random variable} \\ \tau_k \in [0, \infty >: & \text{transmission delay along axon of neuron } k \end{array} \tag{1.5}$$

If $p_k(t) = 1$, an action potential arriving at the synapse at time t is succesful in releasing neuro-transmitter. If $p_k(t) = 0$, it is not.

- Our third approximation is to forget about the spatial extension of the dendrite, assuming all synapses to be located near the cell body, and to treat it as a simple passive cable-like object, the potential of which is reset every time the neuron fires an action potential. This means that the evolution in time of the post-synaptic potential $V(t)$ can be written as a linear differential equation of the form:

$$\frac{d}{dt} V(t) = \frac{d}{dt} V(t)|_{\text{passive}} + \frac{d}{dt} V(t)|_{\text{reset}} \tag{1.6}$$

The first term represents the passive cable-like electric behaviour of the dendrite:

$$\frac{d}{dt} V(t)|_{\text{passive}} = \frac{1}{\tau} [\tilde{I} + \sum_{k=1}^{N} I_k(t) - \rho V(t)] \tag{1.7}$$

in which the two parameters τ and ρ reflect the electrical properties of the dendrite (τ being the characteristic time for current changes to effect the voltage, ρ controlling the stationary ratio between voltage and current), and $\tilde{I}$ represents the stationary currents due to the ion pumps. Without any input from other neurons, i.e. for $I_k(t) = 0$, this term would lead to a simple exponential decay with relaxation time τ/ρ of the voltage towards the stationary value $V(\infty) = \tilde{I}/\rho$. The second term, coming into play only when the neuron has fired recently, represents a quick reset (relaxation time $\Delta \ll \tau/\rho$ towards this stationary value:

$$\frac{d}{dt} V(t)|_{\text{reset}} = \frac{1}{\Delta} [\tilde{I} - \rho V(t)] \; \theta[\int_0^{\Delta} ds \; S(t-s)] \tag{1.8}$$

We may equivalently write $\tilde{I} = \rho V_{\rm rest}$. In combination we obtain:

$$\tau \frac{d}{dt} V(t) = \sum_{k=1}^{N} I_k(t) - \rho[V(t) - V_{\rm rest}] \left\{ 1 + \frac{\tau}{\Delta} \theta[\int_0^{\Delta} ds \; S(t-s)] \right\} \tag{1.9}$$

Since only potential *differences* are important, our equations simplify considerably if, instead of the potential $V(t)$ itself, we write everything in terms of its value relative to the rest potential:

$$V(t) = V_{\rm rest} + U(t) \qquad V^*(t) = V_{\rm rest} + U^*(t) \tag{1.10}$$

This results in

$$\tau \frac{d}{dt} U(t) = \sum_{k=1}^{N} I_k(t) - \rho U(t) \left\{ 1 + \frac{\tau}{\Delta} \theta[\int_0^{\Delta} ds \; S(t-s)] \right\} \tag{1.11}$$

Putting all ingredients together, keeping in mind that the above equations apply to each of the N neurons, and assuming all neurons to be identical in their electrical properties $\{\tau, \rho, V_{\rm rest}\}$, we obtain:

$$\tau \frac{d}{dt} U_i(t) = \sum_{k=1}^{N} J_{ik} p_{ik}(t) S_k(t-\tau_{ik}) - \rho U_i(t) \left\{ 1 + \frac{\tau}{\Delta} \theta[\int_0^{\Delta} ds \; S_i(t-s)] \right\} \tag{1.12}$$

with

$$\begin{array}{ll} J_{ik} \in \langle -\infty, \infty \rangle : & \text{synapse connecting } k \to i \\ \tau_{ik} \in [0, \infty\rangle : & \text{time for signals to travel from } k \to i \\ p_{ik}(t) \in \{0, 1\} : & \text{succes or failure of transmitter release at } k \to i \\ S_k(t) = \theta[U_k(t) - U_k^*(t)] : & \text{firing state of neuron } k \end{array} \tag{1.13}$$

The above equation still describes many of the neuron characteristics which seem relevant. However, it contains the noise variables $\{p_{ij}(t), U_i^*(t)\}$, and, although sufficiently simple to simulate on a computer, it is still too complicated to allow us to proceed analytically.

The next stage in the argument is to work out the effect of the random variables describing transmitter release $\{p_{ij}(t)\}$ and threshold noise $\{U_i^*(t)\}$.

- At each time and each synapse the $p_{ik}(t)$ are assumed to be completely independently distributed (without correlations with potentials, synaptic strengths, thresholds or transmission delays), according to

$$\begin{array}{l} \text{Prob}\,[p_{ij}(t) = 1] = p \\ \text{Prob}\,[p_{ij}(t) = 0] = 1 - p \end{array} \tag{1.14}$$

In particular:

$$\overline{p_{ij}(t)} = p \tag{1.15}$$

$$\overline{p_{ij}(t) p_{kl}(t)} = p\delta_{ik}\delta_{jl} + p^2\,[1 - \delta_{ik}\delta_{jl}] = p^2 + p(1-p)\delta_{ik}\delta_{jl} \tag{1.16}$$

where we have used the Kronecker symbol δ_{ij},

$$\begin{array}{ll} \delta_{ij} = 1 & \text{if } i = j \\ \delta_{ij} = 0 & \text{otherwise} \end{array}$$

So we assume all synapses to be identical in their average failure/success rate.

- Also at each time and each neuron the $U_i^*(t) \in \langle -\infty, \infty \rangle$ are assumed to be completely independently distributed around some average value U_i^*, according to the probability density $P(u)$:

$$\text{Prob}\,[U_i^*(t) - U_i^* \in [u, u+du]] = P(u)du \qquad (0 < du \ll 1) \tag{1.17}$$

In particular:

$$\overline{S_i(t)} = \int_{-\infty}^{\infty} du\ P(u)\theta[U_i(t) - U_i^* - u] = \int_{-\infty}^{U_i(t)-U_i^*} du\ P(u) \tag{1.18}$$

$$\begin{aligned}\overline{S_i(t)S_k(t')} &= \delta_{ik}\overline{S_i(t)\,S_k(t')} + (1-\delta_{ik})\overline{S_i(t)}\ \overline{S_k(t')} \\ &= \overline{S_i(t)}\ \overline{S_k(t')} + \delta_{ik}\left\{\overline{S_i(t)S_k(t')} - \overline{S_i(t)}\ \overline{S_k(t')}\right\}\end{aligned} \tag{1.19}$$

Note that $dt.\overline{S_i(t)} = dt.\int_{-\infty}^{U_i(t)-U_i^*} du\ P(u)$ is precisely the probability that neuron i fires in the infinitesimal time interval $[t, t+dt\rangle$. For situations where the potential $U_i(t)$ does not vary too much wich time (apart from a regular reset due to neuron i itself firing), this probability can be shown to be roughly proportional to the firing frequency $f_i(t)$ of neuron i:

$$f_i(t) = \frac{\text{number of firings in [t}-\text{dt, t}+\text{dt]}}{2dt} \sim \frac{\overline{S_i(t)}.2dt/\Delta}{2dt} = \frac{1}{\Delta}\overline{S_i(t)} \tag{1.20}$$

(taking into account the refractory period Δ).

- For sufficiently large systems, $N \gg 1$, the outcome of the summation over k in equation (1.12) will be described by a Gaussian probability distribution (Central Limit Theorem[4]), average and variance of which are given by:

$$\text{av.} = \overline{\sum_{k=1}^{N} J_{ik}p_{ik}(t)S_k(t-\tau_{ik})} = \sum_{k=1}^{N} J_{ik}\overline{p_{ik}(t)}\ \overline{S_k(t-\tau_{ik})} = p\sum_{k=1}^{N} J_{ik}\overline{S_k(t-\tau_{ik})} \tag{1.21}$$

$$\begin{aligned}\text{var.} &= \overline{\left[\sum_{k=1}^{N} J_{ik}p_{ik}(t)S_k(t-\tau_{ik})\right]^2} - \text{av.}^2 \\ &= \sum_{k,\ell=1}^{N} J_{ik}J_{i\ell}\overline{p_{ik}(t)p_{i\ell}(t)}\ \overline{S_k(t-\tau_{ik})S_\ell(t-\tau_{i\ell})} - \text{av.}^2 \\ &= p^2\sum_{k,\ell=1}^{N} J_{ik}J_{i\ell}\left\{\overline{S_k(t-\tau_{ik})S_\ell(t-\tau_{i\ell})} - \overline{S_k(t-\tau_{ik})}\ \overline{S_\ell(t-\tau_{i\ell})}\right\} \\ &\qquad + p(1-p)\sum_{k=1}^{N} J_{ik}^2\overline{S_k^2(t-\tau_{ik})}\end{aligned}$$

[4]For the CLT to apply, we would in principle have to check whether a few technical conditions are met; having a sum of a large number of independent random variables is in itself not enough. These subtleties we will not deal with here.

$$= p \sum_{k=1}^{N} J_{ik}^2 \left\{ \overline{S_k(t-\tau_{ik})} - p\overline{S_k(t-\tau_{ik})}^2 \right\} \tag{1.22}$$

(note that we have used: $S_k^2(t) = S_k(t)$ for all k and t).

- We now use a crude scaling argument to suggest that for densely interacting networks we can forget about the width of the Gaussian distribution describing the summation over k in equations (1.12). We assume that each neuron receives input from about $\mathcal{N}$ of the N neurons present, where both $N \gg 1$ and $\mathcal{N} \gg 1$. A measure for the uncertainty in the outcome of the summation, due to the randomness, is:

$$\text{relative uncertainty} = \frac{\sqrt{\text{variance}}}{\text{average}}$$

$$= \frac{\sqrt{\sum_{k=1}^{N} J_{ik}^2 \overline{S_k(t-\tau_{ik})} \left\{ 1 - p\overline{S_k(t-\tau_{ik})} \right\}}}{\sqrt{p} \sum_{k=1}^{N} J_{ik} \overline{S_k(t-\tau_{ik})}} \sim \frac{1}{\sqrt{\mathcal{N}}}$$

If the network is sufficiently densely interacting, $\mathcal{N} \to \infty$, then we can apparently replace the summation in (1.12) by its average, and forget about the uncertainty (in this case all that remains of the random variables $\{p_{ij}(t)\}$ describing synaptic operation is the prefactor p). Note, however, that no such reasoning applies to the probability of any individual neuron firing.

By replacing the summation in (1.12) by its average over the noise variables, we now obtain the following simplified dynamic equation:

$$\tau \frac{d}{dt} U_i(t) = p \sum_{k=1}^{N} J_{ik} g\left[U_k(t-\tau_{ik}) - U_k^*\right] - \rho U_i(t) \left\{ 1 + \frac{\tau}{\Delta} \theta\left[\int_0^{\Delta} ds \; S_i(t-s) \right] \right\} \tag{1.23}$$

with

$$g[x] = \int_{-\infty}^{x} du \; P(u)$$

From here we can obtain directly most of the model neurons, used as starting points of analytical studies, as specific limits or approximations. We will concentrate on the most common ones.

A. Graded Response Neurons. Here we forget about the delays, $\tau_{ij} = 0$ for all (i, j), and we forget about the reset term in (1.23), assuming that the fluctuations of the neuron potential due to the reset mechanism are not relevant. Our basic equations now become (after a redefinition of our variables to get rid of irrelevant prefactors):

$$\tau \frac{d}{dt} U_i(t) = \sum_{k=1}^{N} J_{ik} g\left[U_k(t) - U_k^*\right] - \rho U_i(t) \tag{1.24}$$

$$g[U - U^*] = \int_{-\infty}^{U-U^*} du \; P(u) : \qquad \text{proportional to firing frequency} \tag{1.25}$$

Note that the (nonlinear) function $g[x]$ has the following properties , by construction:

$$\begin{aligned} &(i) \quad \lim_{x\to-\infty} g[x] = 0, \ \ \lim_{x\to\infty} g[x] = 1 \\ &(ii) \quad g'[x] \geq 0 \ \forall x \end{aligned} \tag{1.26}$$

Often one adds to (1.24) a noise term, to account for the both the possibility that the randomness in the potentials $U_i(t)$ is not negligible (in cases where the naive scaling argument is wrong) and for the fluctuations due to the reset mechanism. Contrary to what common sense would suggest, we will find that having some degree of noise will often improve the operation of neural networks.

B. McCulloch-Pitts Neurons. Here we forget about delays, reset mechanisms and noise. In the absence of noise the variables $U_k^*(t)$ reduce to fixed numbers U_k^*, i.e. $g[x] = \theta[x]$. We also neglect the time it takes for electric currents to build up the post-synaptic potential: we put $\tau \to 0$ in equation (1.23) (from which the reset term has been eliminated and in which all $\tau_{ij} = 0$). As a result the postsynaptic potential becomes, after elimination of distracting prefactors:

$$U_i(t) = \sum_{k=1}^{N} J_{ik} S_k(t)$$

Since we have now lost, in a sense, the intrinsic clock of the system (all time constants have been thrown out), we have to restore the dynamics by writing the neuron states in terms of the *previous* value of the post-synaptic potential, i.e.

$$S_k(t+\Delta) = \theta[U_k(t) - U_k^*]$$

Time is now discretised in units of the refractory period Δ, and we obtain the so-called McCulloch-Pitts neurons[5]:

$$S_i(t+\Delta) = \theta\left[\sum_{k=1}^{N} J_{ik} S_k(t) - U_i^*\right] \tag{1.27}$$

In spite of their simplicity, we will see that these neurons are already universal, in that the operation of any (finite) Turing machine can be emulated by an appropriately constructed network of such units.

C. Stochastic Binary Neurons. In the case where we do wish to take into account noise in systems with McCulloch-Pitts type neurons, so that the $U_k^*(t)$ are really random, it is often convenient to take them to have the same variance and write them in a form where the average and the variance are explicit:

$$U_i^*(t) = U_i^* - \frac{1}{2} T z_i(t)$$

with

$$\frac{1}{4}T^2 = \overline{[U_k^*(t) - U_k^*]^2} = \int du \ u^2 P(u) \qquad \overline{z_i(t)} = 0 \qquad \overline{z_i^2(t)} = 1$$

[5] In fact the model proposed by McCulloch and Pitts in 1943 was even simpler !

(the reason for this specific notation will become apparent below). The parameter T measures the amount of noise in the system; for $T = 0$ we find the deterministic (noiseless) laws, for $T \to \infty$ the system behaves in a completely random manner. A second convenient translation is to redefine the neuron state variables, such that the two neuron states 'firing' and 'rest' will be represented by the numbers '$\sigma = +1$' and '$\sigma = -1$', respectively, as opposed to '$S = +1$' and '$S = 0$'. This allows us to make use of all kinds of symmetry properties and of a strong similary between neural dynamics and the physics of magnetic materials (although we need not make this connection explicit, it has led the way in many analytical studies). The translation is simple:

$$S_i(t) = \frac{1}{2}[\sigma_i(t) + 1] \qquad U_i^* = \frac{1}{2}[\sum_k J_{ik} - w_i]$$

The stochastic ± 1 version of the McCulloch-Pitts recipe (1.27) thereby becomes

$$S_i(t+\Delta) = \theta\left[\sum_{k=1}^{N} J_{ik}S_k(t) - U_i^* + \frac{1}{2}Tz_i(t)\right]$$

$$\Downarrow$$

$$\frac{1}{2}[\sigma_i(t+\Delta)+1] = \theta\left[\frac{1}{2}\sum_{k=1}^{N} J_{ik}\sigma_k(t) + \frac{1}{2}w_i + \frac{1}{2}Tz_i(t)\right]$$

$$\Downarrow$$

$$\sigma_i(t+\Delta) = \text{ sgn}\,[h_i(t) + Tz_i(t)] \qquad h_i(t) = \sum_{k=1}^{N} J_{ik}\sigma_k(t) + w_i \tag{1.28}$$

with the sign function $\text{sgn}[x] = 2\theta[x] - 1$ (so $\text{sgn}[x] = 1$ for $x > 0$, $\text{sgn}[x] = -1$ for $x < 0$; $\text{sgn}[0]$ is usually defined to be 0). The quantity $h_i(t)$ is called the 'local field'.

The probability to find a neuron state $\sigma_i(t + \Delta)$ can be expressed in terms of the distribution $P(z)$ of the remaining (independent) noise variables $z_i(t)$. For *symmetric* noise distributions, $P(z) = P(-z)\ \forall z$, this probability can be written in a very compact way:

$$\text{Prob}\,[\sigma_i(t+\Delta) = 1] = \text{Prob}\,[h_i(t) + Tz_i(t) > 0] = \textstyle\int_{-h_i(t)/T}^{\infty} dz\ P(z)$$
$$\text{Prob}\,[\sigma_i(t+\Delta) = -1] = \textstyle\int_{-\infty}^{-h_i(t)/T} dz\ P(z) = \int_{h_i(t)/T}^{\infty} dz\ P(z)$$

so that we can combine the two probabilities into the single expression:

$$\text{Prob}\,[\sigma_i(t+\Delta)] = g\,[\sigma_i(t+\Delta)h_i(t)/T] \qquad g[x] = \int_{-\infty}^{x} dz\ P(z) = \frac{1}{2} + \int_0^x dz\ P(z) \tag{1.29}$$

The function $g[x]$ has by construction the following properties:

$$\begin{aligned}
&(i) \quad && g[x] + g[-x] = 1 \\
&(ii) \quad && \lim_{x\to-\infty} g[x] = 0,\ g[0] = \tfrac{1}{2},\ \lim_{x\to\infty} g[x] = 1 \\
&(iii) \quad && g'[x] = P(x):\ g'[x] \geq 0\ \forall x,\ g'[-x] = g'[x]\ \forall x,\ \lim_{x\to\pm\infty} g'[x] = 0
\end{aligned} \tag{1.30}$$

A natural choice for the distribution $P(z)$ of the noise variables z_i is the Gaussian rule

$$P(z) = (2\pi)^{-\frac{1}{2}} e^{-\frac{1}{2}z^2} \qquad \overline{z} = 0,\ \overline{z^2} = 1$$

$$g[x] = \frac{1}{2} + \int_0^x \frac{dz}{\sqrt{2\pi}} e^{-\frac{1}{2}z^2} = \frac{1}{2} + \int_0^{x/\sqrt{2}} \frac{dz}{\sqrt{\pi}} e^{-z^2} = \frac{1}{2}[1 + \mathrm{erf}(x/\sqrt{2})]$$

An alternative, which will simplify considerably many of our subsequent calculations, is to replace the above function $g[x]$ by the following, similar, one:

$$\tilde{g}[x] = \frac{1}{2}[1 + \tanh(x)] \tag{1.31}$$

This function satisfies our general requirements (1.30), and corresponds to the noise distribution

$$\tilde{P}(z) = \frac{1}{2}[1 - \tanh^2(z)]$$

In a network of such units, updates can be effectuated either synchronously (in parallel) or randomly asynchronously (one after the other). In the first case, since all noise variables are independent, the combined probability to find state $\boldsymbol{\sigma}(t+\Delta) = (\sigma_1(t+\Delta), \ldots, \sigma_N(t+\Delta)) \in \{-1, 1\}^N$ equals the product of the individual probabilities:

$$\text{parallel}: \qquad \mathrm{Prob}\,[\boldsymbol{\sigma}(t+\Delta)] = \prod_{i=1}^{N} g\,[\sigma_i(t+\Delta) h_i(t)/T] \tag{1.32}$$

In the second case we have to take into account that only one neuron changes its state at a time. The candidate is drawn at random (each neuron has probability $\frac{1}{N}$ to be a candidate):

$$\text{sequential}: \qquad \text{choose } i \text{ randomly from } \{1, \ldots, N\}; \qquad \mathrm{Prob}\,[\sigma_i(t+\Delta)] = g\,[\sigma_i(t+\Delta) h_i(t)/T] \tag{1.33}$$

D. Coupled Oscillators. One might speculate that, even though the exact details of the action potentials might not be relevant, going to a description involving only firing frequencies is too crude an approximation in that the degree of synchrony with with the neurons fire might be important. This has led to the proposal to study so-called coupled oscillator models. Essentially one builds in the reset mechanism by saying that the potentials are *periodic* functions of underlying *phase variables*:

$$U_i(t) = f[\phi_i(t)] \qquad f[\phi + 2\pi] = f[\phi]$$

If all phases $\phi_i(t)$ are identical (mod 2π), the neurons fire action potentials coherently. In the absence of mutual interaction all neurons are assumed to oscillate at some neuron-specific basic frequency ω_i. Excitatory synapses are assumed to induce a more coherent firing of the two neurons involved; inhibitory synapses are assumed to lead to incoherent firing. The simplest phenomenological model to have such properties is

$$\frac{d}{dt}\phi_i(t) = \omega_i + \sum_{k=1}^{N} J_{ik} \sin[\phi_k(t) - \phi_i(t)] \tag{1.34}$$

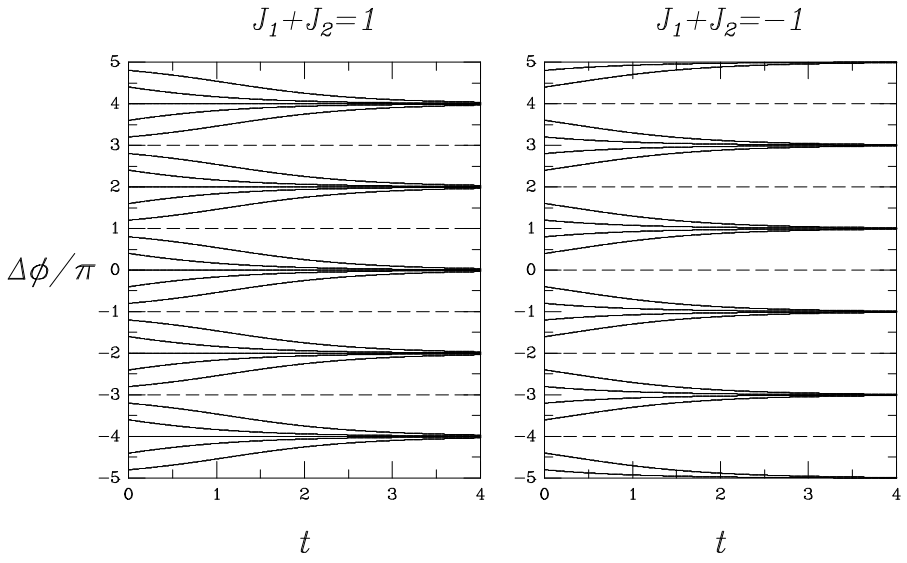


Figure 1.5: Evolution in time of the phase difference $\Delta\phi = \phi_1 - \phi_2$ for $J_1 + J_2 = 1$ (left) and $J_1 + J_2 = -1$ (right). In the first case the two oscillators will always synchronise; their action potentials will eventually be fired simultaneously. In the second case they will do the opposite.

In order to see that this model indeed has the desired properties, we just concentrate on the interaction between a given pair of identical oscillators, say number 1 and number 2 ($\omega_1 = \omega_2$). Here we observe

$$\frac{d}{dt}[\phi_1(t) - \phi_2(t)] = [J_{12} + J_{21}] \sin[\phi_2(t) - \phi_1(t)]$$

The solution of this equation is shown in figure 1.5, for $J_1 + J_2 = 1$ and $J_1 + J_2 = -1$. In the first case the system always evolves towards a synchronised situation, with $\phi_1 - \phi_2 = 2m\pi$ (m integer); in the second case towards an anti-synchronised state, with $\phi_1 - \phi_2 = \pi + 2m\pi$ (m integer). Indeed, for small differences between the two phases, $\phi_1(t) - \phi_2(t) = 2m\pi + \epsilon(t)$ $|\epsilon(t)| \ll 1$, we can linearise this equation:

$$\frac{d}{dt}\epsilon(t) = -[J_{12} + J_{21}]\epsilon(t) + \mathcal{O}(\epsilon^3(t))$$

For $J_{12} + J_{21} > 0$ the phase difference will decrease further (the coherent state $\phi_1(t) = \phi_2(t)$ is stable), whereas for $J_{12} + J_{21} < 0$ the phase difference will increase (now the coherent state $\phi_1(t) = \phi_2(t)$ is unstable). As with the graded response neurons, again one often adds to (1.34) a noise term, to account for the possibility that the randomness in the potentials $U_i(t)$ is not negligible.

1.3 Universality of McCulloch-Pitts Neurons

The simplest neuron model we arrived at was the McCulloch-Pitts neuron (1.27). Here we will show that even this model neuron is universal: the operation of any finite deterministic digital information processing machine can be emulated by a properly constructed network of McCulloch-Pitts neurons. Since the basic logical units of digital machines can all be built with McCulloch-Pitts neurons, all theorems of computer science (on computability, Turing machines etc.) that apply to such digital machines, will apply to networks of McCulloch-Pitts neurons as well. Here we will not dig too deep, and just prove some simple statements.

Reduction to Single-Output Binary Operations. First of all we show how any such machine can be reduced to a collection of simpler single-output binary machines:

- Finite digital machines can only handle finite-precision representations of real numbers, i.e.

  $$\pi \;\rightarrow\; 3.14159265358979323846264 3$$

  Every finite-precision real number can, in turn, be expressed in terms of integers:

  $$\pi \;\rightarrow\; (\; \overbrace{314159265358979323846264 3}^{\text{digits}} \;;\; \overbrace{1}^{\text{decimal point}} \;)$$

  Every integer can, in turn, be expressed as a string of binary numbers $\in \{0,1\}$, e.g.

  $$(10011010111) = 1.2^{10} + 0.2^9 + 0.2^8 + 1.2^7 + 1.2^6 + 0.2^5 + 1.2^4 + 0.2^3 + 1.2^2 + 1.2^1 + 1.2^0$$
  $$= 1024 + 128 + 64 + 16 + 4 + 2 + 1 = 1239$$

- Every finite digital machine can apparently be regarded as mapping finite-dimensional binary input vectors $\boldsymbol{S} \in \Omega \subseteq \{0,1\}^N$ (representing characters, numbers, keys typed on a keyboard, images, whatever) onto finite dimensional binary output vectors $\boldsymbol{S'} \in \{0,1\}^K$ (characters, numbers, reply text on a screen, control signals for other equipment, etc.). Deterministic machines will by definition always reply in exactly the same manner to identical input data. Therefore every finite deterministic digital machine can be specified in full by specifying the output vector $\boldsymbol{S'}(\boldsymbol{S})$ for every possible input vector $\boldsymbol{S} \in \Omega$, i.e. by specifying the mapping M:

  $$M:\; \Omega \rightarrow \{0,1\}^K \qquad M\boldsymbol{S} = \boldsymbol{S'}(\boldsymbol{S})\;\forall \boldsymbol{S} \in \Omega$$

- Finally, we can always construct K independent sub-machines M_ℓ ($\ell = 1, \ldots, K$), each of which takes care of one of the K output bits of the full mapping M:

  $$M_\ell:\; \Omega \rightarrow \{0,1\} \qquad M_\ell\boldsymbol{S} = S'_\ell(\boldsymbol{S})\;\forall \boldsymbol{S} \in \Omega$$

We can therefore concentrate on the question of whether one can perform all of these single-ouput mappings $M : \Omega \subseteq \{0,1\}^N \rightarrow \{0,1\}$ with networks of McCulloch-Pitts neurons.

Reduction to Three Elementary Operations. To do so we proceed to reduce the problem further. We first introduce a partitioning of the set Ω of input vectors into two subsets, depending on the corresponding output value:

$$\Omega = \Omega^+ \cup \Omega^- \qquad \Omega^+ = \{\boldsymbol{S} \in \Omega |\; M\boldsymbol{S} = 1\} \qquad \Omega^- = \{\boldsymbol{S} \in \Omega |\; M\boldsymbol{S} = 0\} \qquad (1.35)$$

We now label the elements in Ω^+. The number of elements in Ω^+, denoted by p, cannot exceed 2^N, which is the total number of vectors in $\{0,1\}^N$.

$$\Omega^+ = \{\boldsymbol{S}^1, \boldsymbol{S}^2, \ldots, \boldsymbol{S}^{p-1}, \boldsymbol{S}^p\} \qquad \boldsymbol{S}^\mu \in \{0,1\}^N$$

Upon introduction of the three logical operators $\wedge$ (AND), $\vee$ (OR) and $\neg$ (NOT), which are defined by their output tables

$\wedge : \{0,1\}^2 \to \{0,1\}$ $\qquad$ $\vee : \{0,1\}^2 \to \{0,1\}$ $\qquad$ $\neg : \{0,1\} \to \{0,1\}$

x	y	$x \wedge y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

x	$\neg x$
0	1
1	0

We can use these basic operations to construct the operator $\mathrm{SAME}(x,y)$, which tells us whether two binary variables have the same value:

$$\mathrm{SAME}(x,y) = (x \wedge y) \vee ((\neg x) \wedge (\neg y))$$

The definitions of AND and OR can be extended to cover more than two argument variables in the usual way:

$$x_1 \wedge x_2 \wedge \cdots \wedge x_{L-1} \wedge x_L = x_1 \wedge (x_2 \wedge (\cdots \wedge (x_{L-1} \wedge x_L) \cdots))$$

$$x_1 \vee x_2 \vee \cdots \vee x_{L-1} \vee x_L = x_1 \vee (x_2 \vee (\cdots \vee (x_{L-1} \vee x_L) \cdots))$$

Since we can write the operation of each single-output operation $M : \Omega \subseteq \{0,1\}^N \to \{0,1\}$ in the form of a look-up exercise, checking whether or not the input vector $\boldsymbol{S}$ is identical to any of the vectors $\boldsymbol{S}^\mu$ in the set Ω^+ (defined above):

$$M\boldsymbol{S} = \Big(\mathrm{SAME}(S_1, S_1^1) \wedge \mathrm{SAME}(S_2, S_2^1) \wedge \cdots \wedge \mathrm{SAME}(S_{N-1}, S_{N-1}^1) \wedge \mathrm{SAME}(S_N, S_N^1)\Big)$$

$$\vee \Big(\mathrm{SAME}(S_1, S_1^2) \wedge \mathrm{SAME}(S_2, S_2^2) \wedge \cdots \wedge \mathrm{SAME}(S_{N-1}, S_{N-1}^2) \wedge \mathrm{SAME}(S_N, S_N^2)\Big)$$

$$\vdots$$

$$\vee \Big(\mathrm{SAME}(S_1, S_1^{p-1}) \wedge \mathrm{SAME}(S_2, S_2^{p-1}) \wedge \cdots \wedge \mathrm{SAME}(S_{N-1}, S_{N-1}^{p-1}) \wedge \mathrm{SAME}(S_N, S_N^{p-1})\Big)$$

$$\vee \Big(\mathrm{SAME}(S_1, S_1^p) \wedge \mathrm{SAME}(S_2, S_2^p) \wedge \cdots \wedge \mathrm{SAME}(S_{N-1}, S_{N-1}^p) \wedge \mathrm{SAME}(S_N, S_N^p)\Big)$$

and since this expression can be constructed completely with the three basic operations $\{\wedge, \vee, \neg\}$, we know that every operation $M : \Omega \subseteq \{0,1\}^N \to \{0,1\}$, and therefore the operation of every finite deterministic digital machine can be reduced to a combination of the these three basic operations.

We can reduce the set of operations further, since the operation $\vee$ (OR) can be written in terms of $\neg$ (NOT) and $\wedge$ (AND), as $x \vee y = \neg((\neg x) \wedge (\neg y))$:

x	y	$x \vee y$	$\neg x$	$\neg y$	$(\neg x) \wedge (\neg y)$
0	0	0	1	1	1
0	1	1	1	0	0
1	0	1	0	1	0
1	1	1	0	0	0

In fact we can even reduce the set of operations further since the operations $\{\wedge, \neg\}$ can. in turn, be written in terms of a single operation NAND (NOT-AND) as $\neg x = \text{NAND}(x, x)$ and $x \wedge y = \text{NAND}(\text{NAND}(x, y), \text{NAND}(x, y))$:

x	y	NAND(x, y)	NAND((NAND(x, y),NAND(x, y))
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

x	NAND(x, x)
0	1
1	0

Elementary Operations via McCulloch-Pitts Neurons. Finally what we have to show is that the elementary operations can be realised with our McCulloch-Pitts neurons

$$S_i(t+\Delta) = \theta\left[\sum_{k=1}^{N} J_{ik} S_k(t) - U_i^*\right]$$

provided we choose appropriate values of the parameters $\{J_{ik}, U_i^*\}$. This we do by construction. Note that in order to prove universality we only have to construct the operation NAND with McCulloch-Pitts neurons, however, by way of illustration we also construct the operations $\{\wedge, \vee, \neg\}$:

x	y	$x \wedge y$	$x + y - 3/2$	$\theta[x + y - 3/2]$
0	0	0	$-3/2$	0
0	1	0	$-1/2$	0
1	0	0	$-1/2$	0
1	1	1	$1/2$	1

x	y	$x \vee y$	$x + y - 1/2$	$\theta[x + y - 1/2]$
0	0	0	$-1/2$	0
0	1	1	$1/2$	1
1	0	1	$1/2$	1
1	1	1	$3/2$	1

x	$\neg x$	$-x + 1/2$	$\theta[-x + 1/2]$
0	1	$1/2$	1
1	0	$-1/2$	0

x	y	NAND(x, y)	$-x - y + 3/2$	$\theta[-x - y + 3/2]$
0	0	1	$3/2$	1
0	1	1	$1/2$	1
1	0	1	$1/2$	1
1	1	0	$-1/2$	0

1.4 A Brief History and Further Reading

Let us conclude with a brief (biased) overview and crude summary of the history of the field of the modelling and analysis of neural information processing. The idea that brain operation and reasoning can in fact be analysed scientifically, with mathematical tools, has for a long time not been a common one. Especially in this field, due to its interdisciplinary nature, it often happened that a new impulse or breakthrough constituted of a discovery or idea that in fact had been proposed already earlier, but at the time went unnoticed.

1888 Discovery of neurons by Ramon y Cajal, using Golgi's staining method (1880). Until this stage there had been two camps: *neuronists* believed the brain consisted of interconnected information processing cells, *reticularists* saw the brain as a continuous uninterrupted network of fibres only.

1937 Turing demystified in a way the concept of 'intelligence' and proved statements about computability by machines. This marked the start of the field of 'Artificial Intelligence'.

1943 McCulloch and Pitts introduced a very simple neuron model, and subsequently proved its universality.

1949 Hebb introduced the idea that biological neural networks store information in the strengths of neural interactions. As a consequence, learning is by definition the modification of interactions, for which Hebb made a proposal.

1961 Rosenblatt and coworkers defined a 'learning rule' according to which a McCulloch-Pitts neuron updates its interaction strengths on basis of examples of input-output relations, corresponding to a task that it is required to perform. They called their systems *perceptrons*, and proved the following 'convergence theorem': if the task is realisable by the perceptron (i.e. if a configuration of interaction strengths exists such that the corresponding perceptron would perform the task perfectly), then the learning rule will converge in a finite number of steps to a configuration that performs the task perfectly.

1969 A thorough mathematical analysis of the potential and restrictions of perceptrons was carried out by Minsky and Papert. One of the things they showed was that single perceptrons could not perform all tasks; some tasks require *layers* of perceptrons, for which no learning rule was known.

1974 Little introduced concepts from the statistical physics of magnetic systems (such as 'temperature'). In fact preceded by Cragg and Temperley in 1954.

1974 Kohonen introduced the idea of building associative memories with recurrent neural networks, where the basic idea is to create specific stable network states by manipulations of the interaction strengths. Earlier proposed by e.g. Taylor in 1956.

1982 Hopfield made the mathematical connection with the statistical mechanics of magnetic systems explicit, through the introduction of an energy function.

1985 The first systematic application of mathematical tools from statistical physics to neural network models was carried out by Amit, Gutfreund and Sompolinsky. They solved explicitly various generalisations of Hopfield's neural network model.

1986 Rumelhart and McClelland derived a learning rule for multi-layerfeed-forward networks of graded-response neurons. Earlier proposed by Werbos in 1974.

1988 Gardner constructed a systematic mathematical procedure to analyse the effect of architectureal constraints on the information processing potential of neural networks.

The last year mentioned being 1988 does not imply a lack of relevant developments since then; it merely reflects the time it takes to clearly see which ideas have an impact and open new doors. At present various complementary tools are used to analyse neural information processing systems, involving techniques and jargon from statistics, information theory, statistical mechanics, computer science, etc.

Some suggestions for further reading are the following books. I have only mentioned those covering subjects that are also in the present course, thereby excluding some more specialised and/or advanced ones.

History:

1. Neurocomputing - Foundations of Research, ed: J.A. Anderson and E. Rosenfeld, 1988, MIT Press. *A book containing reprints of original papers from 1890 to 1987. Compiled from an interdisciplinary perspective, so it does not focus on mathematically oriented papers only.*

2. Perceptrons, M.L. Minsky and S.A. Papert, 1969 (expanded ed. 1988), MIT Press. *The first systematic mathematical analysis of potential and restrictions of a neural network architecture. A nice book with character.*

Textbooks:

1. Introduction to the Theory of Neural Computation, J.A. Hertz, R.G. Palmer and A.S. Krogh, 1991, Addison-Wesley. *A good textbook. Gives a coherent and clear account of common wisdom in this field up until 1989.*

2. An Introduction to the Modeling of Neural Networks, P. Peretto, 1992, Cambridge U.P. *Comparable to the previous one, but here one also finds more on neurobiology and neurocomputers.*

3. Neural Networks: a Comprehensive Foundation, S. Haykin, 1994, MacMillan. *Value for money in terms of volume, but somewhat poor on analysis and biased towards applications.*

Chapter 2

Layered Networks

2.1 Linear Separability

All elementary operations we encountered in the previous chapter could not only be realised with McCulloch-Pitts neurons, but even with a *single*, McCulloch-Pitts neuron. The question naturally arises whether any operation $\{0,1\}^K \to \{0,1\}$ can be performed with a single McCulloch-Pitts neuron. For $K = 1$, the trivial case of only one input variable $x \in \{0,1\}$, we can simply check all possible operations $M : \{0,1\} \to \{0,1\}$ (of which there are four), and see that one can always construct an equivalent McCulloch-Pitts neuron $S(x) = \theta[Jx - U]$:

x	$M_a(x)$	$\theta[-1]$	$M_b(x)$	$\theta[-x+1/2]$	$M_c(x)$	$\theta[x-1/2]$	M_d	$\theta[1]$
0	0	0	1	1	0	0	1	1
1	0	0	0	0	1	1	1	1

For $K > 1$, however, the answer is no. There are several ways of showing this. The simplest is to first give a counterexample for $K = 2$, which can subsequently be used to generate counterexamples for any $K \geq 2$, the XOR operation (exlusive OR):

x	y	XOR(x,y)
0	0	0
0	1	1
1	0	1
1	1	0

Fact: No set of real numbers $\{J_x, J_y, U\}$ exists, such that

$$\theta[J_x x + J_y y - U] = \text{XOR}(x,y) \qquad \forall (x,y) \in \{0,1\}^2$$

Proof: By contradiction. Assume the above statement were false, this would imply:

$$\begin{aligned}
(x,y) = (0,1): &\quad J_y - U > 0 &&\Rightarrow J_y > U \\
(x,y) = (1,0): &\quad J_x - U > 0 &&\Rightarrow J_x > U \\
(x,y) = (1,1): &\quad J_y + J_y - U < 0 &&\Rightarrow J_x + J_y < U
\end{aligned}$$

This contradiction shows that the above statement can never be false, which completes our proof.

For $K > 2$ we can construct a similar operation M by just applying the XOR operation to the first two of the K input variables:

$$M : \{0,1\}^K \to \{0,1\} \qquad M(x_1, \ldots, x_K) = \mathrm{XOR}(x_1, x_2)$$

to which the same proof applies as to the $K = 2$ case discussed above. As a result we know that indeed for all $K \geq 2$ there exist operations that cannot be performed by a single McCulloch-Pitts neuron.

We can also give a geometric picture. The set of all possible operations $M : \{0,1\}^K \to \{0,1\}$ consists of all possible ways to fill the right column of the corresponding table with 1's or 0's:

x_1	x_2	$\cdots$	$\cdots$	x_{K-1}	x_K	$M(\boldsymbol{x})$
0	0	$\cdots$	$\cdots$	0	0	$*$
1	0	$\cdots$	$\cdots$	0	0	$*$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
1	1	$\cdots$	$\cdots$	1	0	$*$
1	1	$\cdots$	$\cdots$	1	1	$*$

The total number of possible operations $M : \{0,1\}^K \to \{0,1\}$ is the number of possible ways to fill the right column of the above table. There are 2^K entries in this column, with two possible values for each, so

$$\text{number of operations } M : \{0,1\}^K \to \{0,1\} \quad = \quad 2^{2^K}$$

The set $\{0,1\}^K$ of possible binary input vectors consists of the corners of the unit hypercube in $\Re^K$. A McCulloch-Pitts neuron, performing the operation

$$S : \{0,1\}^K \to \{0,1\} \qquad S(\boldsymbol{x}) = \theta[\sum_{k=1}^{K} J_k x_k - U]$$

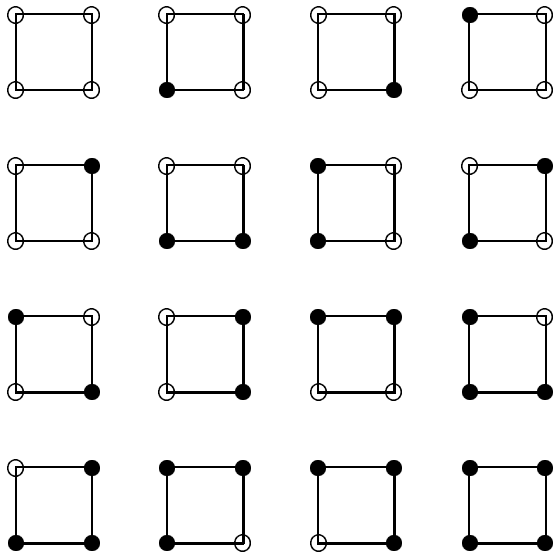
can be seen as dividing the space $\Re^K$ in two subspaces which are separated by the hyperplane $\sum_{k=1}^K J_k x_k = U$. The information conveyed by the outcome of the operation of a McCulloch-Pitts neuron, given an input $\boldsymbol{x} \in \{0,1\}^K$, is simply in which of the two subspaces the corner $\boldsymbol{x}$ of the hypercube is located. $S(\boldsymbol{x}) = 1$ if $\boldsymbol{x}$ is in the subspace $\sum_{k=1}^K J_k x_k > U$; $S(\boldsymbol{x}) = 0$ if $\boldsymbol{x}$ is in the subspace $\sum_{k=1}^K J_k x_k < U$ (let us not yet worry about the pathological case where $\boldsymbol{x}$ is *on* the separating plane for the moment). Note that this division of $\{0,1\}^K$ into subsets, here separated by a plane, is what we already introduced earlier through the subsets Ω^+ and Ω^-. In terms of the above table, Ω^+ is the set of all hypercube corners $\boldsymbol{x} \in \{0,1\}^K$ for which $* = 1$; Ω^- is the set of corners $\boldsymbol{x} \in \{0,1\}^K$ with $* = 0$. Since each of these 2^{2^K} operations is characterised uniquely by the set Ω^+, each can be pictured uniquely by drawing the hypercube in $\Re^K$ in which corners in Ω^+ are coloured black, and corners in Ω^- are white.

For $K = 1$ the hypercube reduces to a line segment with $2^1 = 2$ 'corners'. There are $2^{2^1} = 4$ operations $M : \{0,1\} \to \{0,1\}$, i.e. 4 ways of colouring the two corners:

$$K = 1 : \qquad \boldsymbol{x} \in \{0,1\}, \qquad \begin{array}{ll} \bullet : & \boldsymbol{x} \in \Omega^+, \quad M(\boldsymbol{x}) = 1 \\ \circ : & \boldsymbol{x} \in \Omega^-, \quad M(\boldsymbol{x}) = 0 \end{array}$$

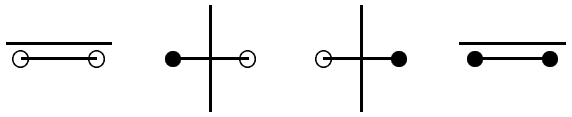
For $K = 2$ the hypercube is a square (with $2^2 = 4$ corners). There are $2^{2^2} = 16$ operations $M : \{0,1\}^2 \to \{0,1\}$, i.e. 16 ways of colouring the four corners:

$$K = 2 : \qquad \boldsymbol{x} \in \{0,1\}^2, \qquad \begin{array}{ll} \bullet : & \boldsymbol{x} \in \Omega^+, \quad M(\boldsymbol{x}) = 1 \\ \circ : & \boldsymbol{x} \in \Omega^-, \quad M(\boldsymbol{x}) = 0 \end{array}$$

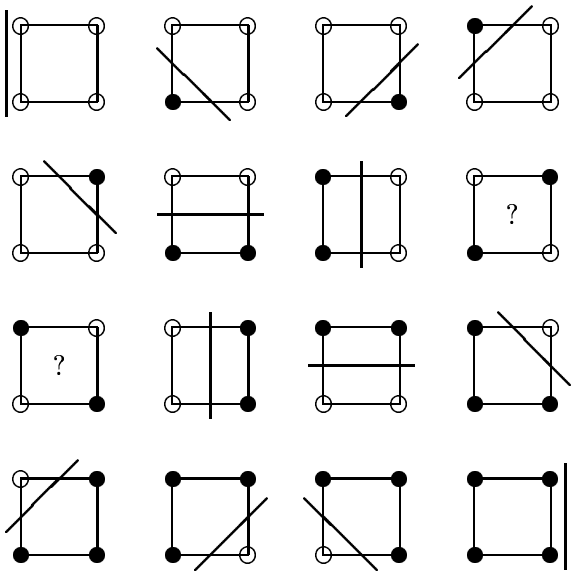


For $K = 3$ the hypercube is a cube (with eight corners). There are $2^{2^3} = 256$ operations $M : \{0,1\}^3 \to \{0,1\}$, i.e. 256 ways of colouring the eight corners, etc. Of all these operations, McCulloch-Pitts neurons can only perform those for which the Ω^+ corners $\boldsymbol{x}$ (the black ones), can be separated from the Ω^- corners $\boldsymbol{x}$ (the white ones) with a *single* plane $\boldsymbol{x} \cdot \boldsymbol{J} = U$. Such operations are called *linearly separable*. It is now very clear for the above examples $K = 1$ and $K = 2$, simply by looking at the graphs, which operations are linearly separable:

$K = 1$:



$K = 2$:



For $K = 1$ we recover our result that all operations are linearly separable (i.e. can be performed by a suitably constructed McCulloch-Pitts neuron). For $K = 2$ we find that of the 16 possible operations, two are not linearly separable:

x	y	$M_a(x,y)$
0	0	0
0	1	1
1	0	1
1	1	0

x	y	$M_b(x,y)$
0	0	1
0	1	0
1	0	0
1	1	1

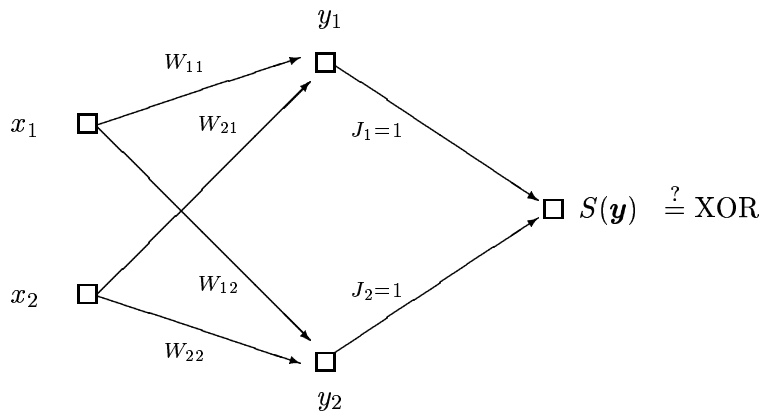
in which we recognise the two operations XOR (as should be) and ¬(XOR).

2.2 Multi-Layer Networks

Having seen that a single McCulloch-Pitts neuron cannot realise every mapping $M : \{0,1\}^K \to \{0,1\}$, we now set out to demonstrate that every such mapping M can at least be performed by a two-layer feed-forward network of such neurons, with only one neuron in the second layer. This will in fact be a specific neural realisation of the look-up table for elements of the set Ω^+, introduced in demonstrating the universality of McCulloch-Pitts neurons.

As a simple explicit example of the 'grandmother' neuron construction, we will now illustrate how a multi-layer Perceptron can be designed to perform a linearly non-separable task such as XOR. Here the set of vectors Ω^+, for which the XOR operation is to give as output the value 1, is

$$\Omega^+ = \{\boldsymbol{x}^1, \boldsymbol{x}^2\} \qquad \text{with} \qquad \boldsymbol{x}^1 = (1,0) \qquad \boldsymbol{x}^2 = (0,1)$$



The 'grandmother' construction is based on our knowledge of Ω^+. For each of the vectors $\boldsymbol{x}$ in Ω^+ there is to be precisely one neuron in the 'hidden' layer with the task to ensure the state of the output neuron to become $+1$ in the case where $\boldsymbol{x}$ occurs as input. Weights W_{ij} and firing thresholds V for which this will be achieved are given by the recipe

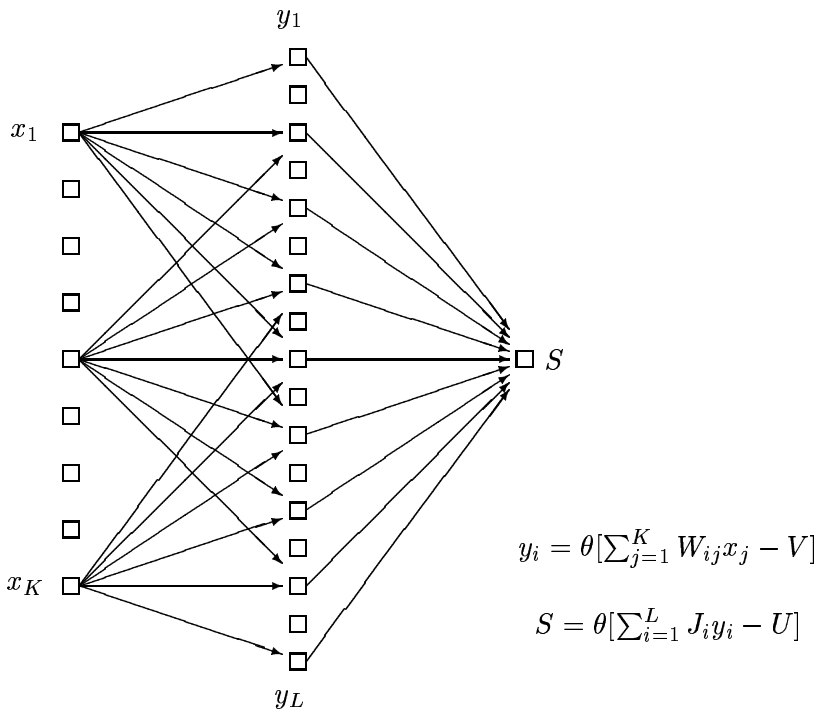
$$\begin{array}{ll} W_{11} = 2(2x_1^1 - 1) = +2 & W_{12} = 2(2x_2^1 - 1) = -2 \\ W_{21} = 2(2x_1^2 - 1) = -2 & W_{22} = 2(2x_2^2 - 1) = +2 \end{array} \qquad V = 2\sum_{j=1}^{2} x_j^1 - 1 = 1$$

Having set the values of our parameters to the above vaules, one can check that the XOR operation is now realised:

$$\begin{array}{ll} \boldsymbol{x} = (0,0): & S(\boldsymbol{y}) = \theta\left[y_1 + y_2 - 1/2\right] = \theta\left[0 + 0 - 1/2\right] = 0 \\ \boldsymbol{x} = (1,0): & S(\boldsymbol{y}) = \theta\left[y_1 + y_2 - 1/2\right] = \theta\left[1 + 0 - 1/2\right] = 1 \\ \boldsymbol{x} = (0,0): & S(\boldsymbol{y}) = \theta\left[y_1 + y_2 - 1/2\right] = \theta\left[0 + 1 - 1/2\right] = 1 \\ \boldsymbol{x} = (1,1): & S(\boldsymbol{y}) = \theta\left[y_1 + y_2 - 1/2\right] = \theta\left[0 + 0 - 1/2\right] = 0 \end{array}$$

We next give the general construction, which works for arbitrary input dimensions and arbitrary operations, which then *en passant* proves that any tramnsformation can be performed by a two-layer feed-forward network. We choose L to be the size of the set $\Omega^+ = \{\boldsymbol{x} \in \{0,1\}^K \mid M(\boldsymbol{x}) = 1\}$, and construct the neurons y_i in the so-called 'hidden' layer such that the operation of each of them is defined as determining whether the input vector $\boldsymbol{x}$ equals a neuron-specific prototype vector from Ω^+. Such so-called 'grandmother neurons' are constructed easily. First we define the building block G:

$$\boldsymbol{x}, \boldsymbol{x}^* \in \{0,1\}^K: \qquad G[\boldsymbol{x}^*; \boldsymbol{x}] = \theta\left[\sum_{i=1}^{K} (2x_i - 1)(2x_i^* - 1) - K + 1\right] \tag{2.1}$$



Fact: $G[\boldsymbol{x}^*; \boldsymbol{x}] = 1$ if and only if $\boldsymbol{x} = \boldsymbol{x}^*$.

Proof: We first turn to the trivial 'if' part of the proof:

$$G[\boldsymbol{x}; \boldsymbol{x}] = \theta \left[\sum_{i=1}^{K} (2x_i - 1)^2 - K + 1 \right] = \theta \left[\sum_{i=1}^{K} 1 - K + 1 \right] = 1$$

Next we show that $G[\boldsymbol{x}^*; \boldsymbol{x}] = 0$ as soon as $\boldsymbol{x} \neq \boldsymbol{x}^*$.

$$\forall x_i, x_i^* \in \{0, 1\} : \quad (2x_i - 1)(2x_i^* - 1) = \begin{cases} 1 & \text{if } x_i = x_i^* \\ -1 & \text{if } x_i \neq x_i^* \end{cases}$$

Consequently:

$$\sum_{i=1}^{K} (2x_i - 1)(2x_i^* - 1) = K - 2 \times (\text{number of indices } i \text{ with } x_i \neq x_i^*)$$

so that:

$$\boldsymbol{x} \neq \boldsymbol{x}^* \;\Rightarrow\; \sum_{i=1}^{K} (2x_i - 1)(2x_i^* - 1) \leq K - 2 \;\Rightarrow\; G[\boldsymbol{x}^*; \boldsymbol{x}] = 0$$

which completes the proof.

The expression $G[\boldsymbol{x}^*; \boldsymbol{x}]$, interpreted as an operation performed on the variable $\boldsymbol{x}$ (for fixed $\boldsymbol{x}^*$), is clearly of the McCulloch-Pitts form:

$$G[\boldsymbol{x}^*; \boldsymbol{x}] = \theta \left[2 \sum_{i=1}^{K} (2x_i^* - 1) x_i - 2 \sum_{i=1}^{K} x_i^* + 1 \right]$$

We can now construct our two-layer network from these building blocks.

$$\Omega^+ = \{\boldsymbol{x} \in \{0,1\}^K |\ M(\boldsymbol{x}) = 1\} = \{\boldsymbol{x}^1, \boldsymbol{x}^2, \ldots, \boldsymbol{x}^{L-1}, \boldsymbol{x}^L\}$$

$$\forall i \in \{1, \ldots, L\}: \qquad y_i : \{0,1\}^K \to \{0,1\}, \quad y_i = G[\boldsymbol{x}^i; \boldsymbol{x}]$$

The required operation of the output neuron S, finally, is to simply detect whether any of the hidden layer neuron states y_i equals one. Our final result is the following network:

$$y_i(\boldsymbol{x}) = \theta[\sum_{j=1}^{K} W_{ij} x_j - V] \qquad S(\boldsymbol{y}) = \theta[\sum_{i=1}^{L} y_i - \frac{1}{2}] \tag{2.2}$$

$$W_{ij} = 2(2x_j^i - 1), \quad V = 2\sum_{j=1}^{K} x_j^i - 1 \tag{2.3}$$

This construction will exactly perform the required operation. The state of each hidden neuron states whether ($y_i = 1$) or not ($y_i = 0$) element number i of the set Ω^+ equals the actual input $\boldsymbol{x}$; output neuron S subsequently tells us whether ($S = 1$) or not ($S = 0$) there is a $+1$ state in the hidden layer, i.e. whether $\boldsymbol{x}$ is in the set Ω^+.

Since the size of Ω^+ can scale exponentially in K, this is usually not an efficient way of realising the operation M with McCulloch-Pitts neurons, but that was not our aim. At least we know now that a two-layer feedforward architecture is capable of realising any operation, if properly contructed.

2.3 The Perceptron

Here we turn to the question of how to model and understand the process of 'learning'. In the relatively simple context of a McCulloch-Pitts neuron $S(\boldsymbol{x}) = \theta[\boldsymbol{J} \cdot \boldsymbol{x} - U]$ this means the adaptation of the set of connections $\{J_i\}$ and the threshold U, in order to improve the accuracy in the execution of a certain task $M : \Omega \subseteq \{0,1\}^K \to \{0,1\}$. We assume that we do not know the task M explicitly; we only have examples given by some 'teacher' of 'questions' (input vectors $\boldsymbol{x} \in \{0,1\}^K$) with corresponding 'answers' (the output values $M(\boldsymbol{x})$).

The starting point of the training session is a neuron with, say, randomly choosen parameters $\{J_i\}$ and U. A so-called 'on-line' training session consists of iterating the following procedure:

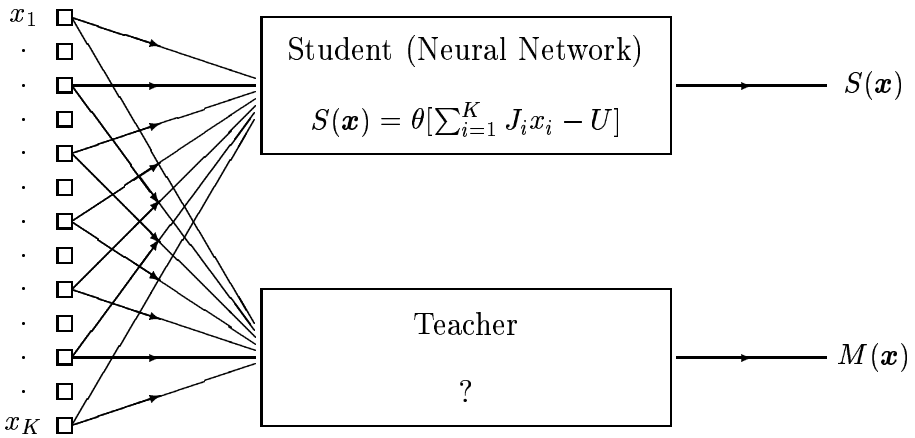
step 1: Draw at random a question $\boldsymbol{x} \in \Omega$

step 2: Check whether teacher and student agree on the answer:

$$\begin{array}{ll} M(\boldsymbol{x}) = S(\boldsymbol{x}): & \text{do nothing, return to 1} \\ M(\boldsymbol{x}) \neq S(\boldsymbol{x}): & \text{modify parameters, then return to 1} \end{array}$$

modify parameters:

$$\begin{array}{rcl} \boldsymbol{J} & \to & \boldsymbol{J} + \Delta\boldsymbol{J}[\boldsymbol{J}, U; \boldsymbol{x}, M(\boldsymbol{x})] \\ U & \to & U + \Delta U[\boldsymbol{J}, U; \boldsymbol{x}, M(\boldsymbol{x})] \end{array}$$



The input vectors $\boldsymbol{x}$ need not all have the same probability of being drawn. The ultimate goal is to end up with a neuron that has 'learned' to perform the task M perfectly, i.e. $M(\boldsymbol{x}) = S(\boldsymbol{x}) = \theta[\boldsymbol{J} \cdot \boldsymbol{x} - U]$ $\forall \boldsymbol{x} \in \Omega$ (which, of course, is only possible if the task M is itself linearly separable). The problem is how to choose an appropriate recipe $(\Delta \boldsymbol{J}, \Delta U)$ for updating the neuron's parameters that will achieve this.

A Perceptron is defined as a McCulloch-Pitts neuron, learning in an on-line fashion, according to the following rule for updating its parameters:

$$\text{Perceptron Learning Rule :} \qquad \begin{cases} M(\boldsymbol{x}) = 0,\ S(\boldsymbol{x}) = 1: & \Delta \boldsymbol{J} = -\boldsymbol{x}, & \Delta U = 1 \\ M(\boldsymbol{x}) = 1,\ S(\boldsymbol{x}) = 0: & \Delta \boldsymbol{J} = \boldsymbol{x}, & \Delta U = -1 \end{cases} \tag{2.4}$$

This recipe is rather transparent. If $S(\boldsymbol{x}) = 1$, but should have been 0, the effect of the modification is to *decrease* the local field $h = \boldsymbol{J} \cdot \boldsymbol{x} - U$ so that the next time question $\boldsymbol{x}$ appears the Perceptron is more likely to give the (correct) answer $S(\boldsymbol{x}) = 0$. Conversely, if $S(\boldsymbol{x}) = 0$ but should have been 1, the modification causes an *increase* the local field so that the Perceptron is more likely to give the (correct) answer $S(\boldsymbol{x}) = 1$ in the future.

Perceptron Convergence Theorem. The nice and powerful property of this specific recipe is the existence of the following Perceptron Convergence Theorem:

Fact: If the task M is linearly separable, then the above procedure will converge in a finite number of modification steps to a stationary configuration, where $\forall \boldsymbol{x} \in \Omega :\ S(\boldsymbol{x}) = M(\boldsymbol{x})$.

Proof: We first simplify our equations by introducing an additional dummy input variable, which is simply constant: $x_0 = 1$. This allows us, together with the identification $J_0 = -U$, to write the perceptron and its learning rule in the compact form

$$S(\boldsymbol{x}) = \theta[\boldsymbol{J} \cdot \boldsymbol{x}], \qquad \boldsymbol{x} = (x_0, x_1, \ldots, x_K) \in \{0,1\}^{K+1}, \quad \boldsymbol{J} = (J_0, J_1, \ldots, J_K) \in \Re^{K+1}$$

$$\text{Learning Rule :} \qquad \begin{array}{ll} M(\boldsymbol{x}) = 0,\ S(\boldsymbol{x}) = 1: & \Delta \boldsymbol{J} = -\boldsymbol{x} \\ M(\boldsymbol{x}) = 1,\ S(\boldsymbol{x}) = 0: & \Delta \boldsymbol{J} = \boldsymbol{x} \end{array} \iff \Delta \boldsymbol{J} = [2M(\boldsymbol{x}) - 1]\boldsymbol{x}$$

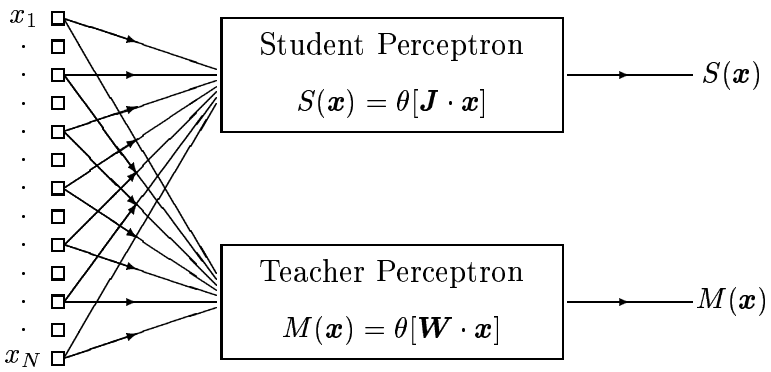
The new set Ω now consists of all vectors $(x_0, x_1, \ldots, x_K)$ with $x_0 = 1$ and with $(x_1, \ldots, x_K)$ in the original input set.

- The first step of the proof is to translate the linear separability of the operation M into the following statement:

$$\exists \boldsymbol{W} \in \Re^{K+1} \text{ such that } \forall \boldsymbol{x} \in \Omega : \; M(\boldsymbol{x}) = \theta[\boldsymbol{W} \cdot \boldsymbol{x}]$$

and, since the argument of the step function can therefore never be zero:

$$\exists \delta > 0 : \text{ such that } \forall \boldsymbol{x} \in \Omega : \; |\boldsymbol{W} \cdot \boldsymbol{x}| > \delta$$



We may consequently view the task operation M as generated by a 'teacher perceptron', equipped with weights/threshold $\boldsymbol{W}$. Since $|\boldsymbol{W}|$ can be rescaled arbitrarily without affecting the associated operation M, we may choose $|\boldsymbol{W}| = 1$.

- A modification step, where $\boldsymbol{J} \to \boldsymbol{J}' = \boldsymbol{J} + \Delta\boldsymbol{J}$, obeys the following two inequalities:

$$\begin{aligned} \boldsymbol{J}' \cdot \boldsymbol{W} &= \boldsymbol{J} \cdot \boldsymbol{W} + [2M(\boldsymbol{x}) - 1]\boldsymbol{x} \cdot \boldsymbol{W} \\ &= \boldsymbol{J} \cdot \boldsymbol{W} + [2\theta[\boldsymbol{x} \cdot \boldsymbol{W}] - 1]\boldsymbol{x} \cdot \boldsymbol{W} \\ &= \boldsymbol{J} \cdot \boldsymbol{W} + |\boldsymbol{x} \cdot \boldsymbol{W}| \\ &\geq \boldsymbol{J} \cdot \boldsymbol{W} + \delta \end{aligned} \tag{2.5}$$

and

$$\begin{aligned} |\boldsymbol{J}'|^2 &= [\boldsymbol{J} + [2M(\boldsymbol{x}) - 1]\boldsymbol{x}]^2 \\ &= \boldsymbol{J}^2 + 2 \; \mathrm{sgn}[\boldsymbol{W} \cdot \boldsymbol{x}]\boldsymbol{J} \cdot \boldsymbol{x} + \boldsymbol{x}^2 \\ &\leq \boldsymbol{J}^2 - 2 \; \mathrm{sgn}[\boldsymbol{J} \cdot \boldsymbol{x}]\boldsymbol{J} \cdot \boldsymbol{x} + K + 1 \\ &\leq \boldsymbol{J}^2 + K + 1 \end{aligned} \tag{2.6}$$

After n such modification steps, repeated usage of the above inequalities gives:

$$\boldsymbol{J}(n) \cdot \boldsymbol{W} \geq \boldsymbol{J}(0) \cdot \boldsymbol{W} + n\delta \qquad |\boldsymbol{J}(n)|^2 \leq |\boldsymbol{J}(0)|^2 + n(K+1) \tag{2.7}$$

- We now define
$$\omega = \frac{\boldsymbol{J} \cdot \boldsymbol{W}}{|\boldsymbol{J}|}$$
Due to the Schwarz inequality $|\boldsymbol{x} \cdot \boldsymbol{y}| \leq |\boldsymbol{x}||\boldsymbol{y}|$ we know that $|\omega| \leq |\boldsymbol{W}| = 1$. After n modifications we have
$$\omega(n) = \frac{\boldsymbol{J}(n) \cdot \boldsymbol{W}}{|\boldsymbol{J}(n)|} \geq \frac{\boldsymbol{J}(0) \cdot \boldsymbol{W} + n\delta}{\sqrt{|\boldsymbol{J}(0)|^2 + n(K+1)}} \tag{2.8}$$
From this we conclude that there can be only a *finite* number of modification steps n, the algorithm will have to stop at some stage, since otherwise we would run into a conflict with $|\omega(n)| \leq 1$:
$$\lim_{n\to\infty} \omega(n) \geq \lim_{n\to\infty} \frac{\boldsymbol{J}(0) \cdot \boldsymbol{W} + n\delta}{\sqrt{|\boldsymbol{J}(0)|^2 + n(K+1)}} = \infty$$
If no more modifications can take place, the system must by definition be in a configuration where $M(\boldsymbol{x}) = S(\boldsymbol{x})\ \forall \boldsymbol{x} \in \Omega$, which completes the proof.

The perceptron convergence theorem is a very strong statement, as it depends only on the linear separability of the task M being learned. In particular it doesn't depend on the probability distribution of the input vectors $\boldsymbol{x}$. An upper bound $n_{\max}$ for the number n of modification steps required can be obtained by checking at what stage we actually run into conflict with $|\omega(n)| \leq 1$:
$$\frac{[\boldsymbol{J}(0) \cdot \boldsymbol{W} + n_{\max}\delta]^2}{|\boldsymbol{J}(0)|^2 + n_{\max}(K+1)} = 1$$
For zero initial connections, $\boldsymbol{J}(0) = (0, \ldots, 0)$, we obtain:
$$n_{\max} = \frac{K+1}{\delta^2}$$
Although this is of limited practical value, as we usually have no information about δ, it is consistent with our geometrical picture of the linearly separable operations: the more complicated the operation M, the closer the separating plane will be to the corners of the hypercube, the smaller δ, and therefore the larger the number of adaptation steps the perceptron needs to obtain perfect performance.

Note that the number n is not the same as the number of times we have to present an example input vector $\boldsymbol{x}$, but the number of actual parameter modifications. It could happen that the number of times we need to draw at random a question $\boldsymbol{x}$ is still very large, simply due to the small probability of some particular relevant questions to be selected. Alternatively, we could also draw the example vectors $\boldsymbol{x}$ in a fixed order, since the perceptron convergence theorem does not require them to be drawn at random. In the latter case the number of times we present a question $\boldsymbol{x}$ will also be bounded. However, since, as we will see later, the number of randomly drawn questions $\boldsymbol{x} \in \{0,1\}^K$ needed to obtain convergence (for a linearly separable task) scales linearly with K for large K, it is usually more efficient to draw the examples at random (note: making a single full sweep through the set $\{0,1\}^K$ of possible questions involves 2^K trials).

Ising Perceptrons. The only property of the input vector space $\Omega \subseteq \{0,1\}^K$ that is actually used in proving convergence of the perceptron learning rule is the existence of an upper bound

for the norm of $\boldsymbol{x}$:

$$\exists C: \text{ such that } \forall \boldsymbol{x} \in \Omega: \ \boldsymbol{x}^2 \leq C$$

(the constant C will then simply replace the factor $K+1$ in equation (2.7)). In particular, we can therefore apply the same procedure to $\boldsymbol{x} \in \Omega \subseteq \{-1,1\}^N$. If we also transform the perceptron output and the task definition in the familiar way from the $\{0,1\}$ representation to the so-called 'Ising spin' variables $\{-1,1\}$, we obtain the Ising perceptron:

$$\begin{array}{ll} \text{task}: & T: \Omega \subseteq \{-1,1\}^N \to \{-1,1\} \\ \text{Ising perceptron}: & \sigma: \Omega \subseteq \{-1,1\}^N \to \{-1,1\}, \quad \sigma(\boldsymbol{x}) = \ \text{sgn}[\boldsymbol{J}\cdot\boldsymbol{x} + w] \end{array} \tag{2.9}$$

with the learning rule

step 1: Draw at random a question $\boldsymbol{x} \in \Omega$

step 2: Check whether teacher and student agree on the answer:

$$\begin{array}{lll} T(\boldsymbol{x}) = \sigma(\boldsymbol{x}): & \text{do nothing} & \text{return to 1} \\ T(\boldsymbol{x}) \neq \sigma(\boldsymbol{x}): & \Delta\boldsymbol{J} = T(\boldsymbol{x})\boldsymbol{x}, \ \Delta w = T(\boldsymbol{x}) & \text{then return to 1} \end{array}$$

Note that this learning procedure (as the original $\{0,1\}$ version) can be written in an even more compact way, without the explicit check on whether or not perceptron and teacher give the same answer to a given question $\boldsymbol{x}$. With the convention $x_0 = 1 \ \forall \boldsymbol{x} \in \Omega$ and $J_0 = w$ (the dummy variable x_0 as before takes care of the threshold) we may write:

$$\text{Draw at random an } \ \boldsymbol{x} \in \Omega, \qquad \Delta\boldsymbol{J} = \frac{1}{2}\left[T(\boldsymbol{x}) - \ \text{sgn}(\boldsymbol{J}\cdot\boldsymbol{x})\right]\boldsymbol{x} \tag{2.10}$$

We can now run through the perceptron convergence proof. We simply replace $M(\boldsymbol{x}) \to \frac{1}{2}[T(\boldsymbol{x})+1]$ and use for the extended input vector $\boldsymbol{x} \in \{-1,1\}^{N+1}$ the inequality $\boldsymbol{x}^2 \leq N+1$.

The advantage of the $\{-1,1\}$ formulation over the previous $\{0,1\}$ one is a significant simplification of subsequent calculations. For instance, for randomly and uniformly drawn vectors $\boldsymbol{x} = (x_1, \ldots, x_N)$ we find the expectation values:

$$\begin{array}{lll} \boldsymbol{x} \in \{0,1\}^N & \langle x_i \rangle = \frac{1}{2} & \langle x_i x_j \rangle = \frac{1}{4} + \frac{1}{4}\delta_{ij} \\ \boldsymbol{x} \in \{-1,1\}^N & \langle x_i \rangle = 0 & \langle x_i x_j \rangle = \delta_{ij} \end{array}$$

The Continuous Time Limit. With the familiar convention $x_0 = 1 \ \forall \boldsymbol{x} \in \Omega$ we can write the learning rule of the original $\{0,1\}$ representation of the perceptron in the compact form

$$\text{Draw at random an } \ \boldsymbol{x} \in \Omega, \qquad \Delta\boldsymbol{J} = \epsilon\boldsymbol{x}\left[M(\boldsymbol{x}) - \theta(\boldsymbol{J}\cdot\boldsymbol{x})\right] \tag{2.11}$$

without the need for checking explicitly whether $M(\boldsymbol{x}) = \theta(\boldsymbol{J}\cdot\boldsymbol{x})$. We have inserted a prefactor ϵ, with $0 < \epsilon \ll 1$, which controls the rate at which the parameters $\boldsymbol{J}$ change. Note that with this ϵ the perceptron convergence theorem still applies. We will derive, in a specific limit, a differential equation to describe the evolution in time of the parameters $\boldsymbol{J}$. A

more thorough derivation requires elements from the theory of stochastic processes; here we will restrict ourselves to a reasonably simple derivation. The resulting differential equation is considerably more convenient to describe the learning process than the above iterative map (2.11).

We now measure our time in units of ϵ, so

$$\Delta \boldsymbol{J} = \boldsymbol{J}(t+\epsilon) - \boldsymbol{J}(t)$$

After a modest number $n \ll \epsilon^{-1}$ of iteration steps the vector $\boldsymbol{J}$ will have changed only a little,

$$\boldsymbol{J}(t+n\epsilon) = \boldsymbol{J}(t) + \mathcal{O}(n\epsilon)$$

For intermediate stages $\ell < n$ we may therefore write:

$$\boldsymbol{J}(t+\ell\epsilon+\epsilon) - \boldsymbol{J}(t+\ell\epsilon) = \epsilon \boldsymbol{x}_\ell \left[M(\boldsymbol{x}_\ell) - \theta(\boldsymbol{J}(t)\cdot\boldsymbol{x}_\ell + \mathcal{O}(n\epsilon))\right]$$

in which $\boldsymbol{x}_\ell$ denotes the input vector that is drawn from Ω at iteration step ℓ. We now sum the left- and the right-hand side of this equation over the iteration index $\ell = 0, \ldots, n-1$:

$$\sum_{\ell=0}^{n-1} \boldsymbol{J}(t+\ell\epsilon+\epsilon) - \sum_{\ell=0}^{n-1} \boldsymbol{J}(t+\ell\epsilon) = \epsilon \sum_{\ell=0}^{n-1} \boldsymbol{x}_\ell \left[M(\boldsymbol{x}_\ell) - \theta(\boldsymbol{J}(t)\cdot\boldsymbol{x}_\ell + \mathcal{O}(n\epsilon))\right]$$

which gives

$$\frac{\boldsymbol{J}(t+n\epsilon) - \boldsymbol{J}(t)}{n\epsilon} = \frac{1}{n} \sum_{\ell=0}^{n-1} \boldsymbol{x}_\ell \left[M(\boldsymbol{x}_\ell) - \theta(\boldsymbol{J}(t)\cdot\boldsymbol{x}_\ell + \mathcal{O}(n\epsilon))\right]$$

We now take a limit where $n\epsilon \to 0$ and $n \to \infty$ (e.g. $n = \epsilon^{-\gamma}$ with $0 < \gamma < 1$). As a result the left-hand side of the above equation becomes a temporal derivative; in the right-hand side we obtain an average over the input set Ω:

$$\frac{d}{dt}\boldsymbol{J}(t) = \langle \boldsymbol{x} \left[M(\boldsymbol{x}) - \theta(\boldsymbol{J}(t)\cdot\boldsymbol{x})\right] \rangle_\Omega \tag{2.12}$$

with $\langle f(\boldsymbol{x}) \rangle_\Omega = \sum_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x}) f(\boldsymbol{x})$ and with $p(\boldsymbol{x})$ denoting the probability that input vector $\boldsymbol{x}$ is drawn from Ω during the learning process. This equation (2.12) is valid for times measured in units of the learning rate ϵ, in the limit $\epsilon \to 0$.

If we apply this derivation to the Ising perceptron (2.9,2.10), we find the same continuous time equation for $\boldsymbol{J}(t)$, now written in the form

$$\frac{d}{dt}\boldsymbol{J}(t) = \frac{1}{2} \langle \boldsymbol{x} \left[T(\boldsymbol{x}) - \operatorname{sgn}(\boldsymbol{J}(t)\cdot\boldsymbol{x})\right] \rangle_\Omega \tag{2.13}$$

At first sight it might seem that the continuous time equation is weaker than the discrete time one, as it is derived in a specific temporal limit and involves only averages over the distribution Ω. Nevertheless we can still prove that, provided the task is linearly separable, the continuous time equation will converge towards the desired state. This can even be shown

to happen in a finite time. We restrict ourselves for simplicity[1] to discrete and finite input sets Ω, and we define the constant $K > 0$ as

$$K = \min_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x})|\boldsymbol{W}\cdot\boldsymbol{x}|$$

Fact: If the task M is linearly separable, then equation (2.12) will converge to a fixed-point $\boldsymbol{J}$ such that $\theta(\boldsymbol{J}\cdot\boldsymbol{x}) = M(\boldsymbol{x})\ \forall\boldsymbol{x}\in\Omega$.

Proof: First we show that the dynamic equation (2.12) must lead to a fixed-point. Then we show that this fixed-point corresponds to a state where the perceptron performs the task M perfectly.

- Since M is linearly separable, $\exists\boldsymbol{W}$ such that $M(\boldsymbol{x}) = \theta(\boldsymbol{W}\cdot\boldsymbol{x})\ \forall\boldsymbol{x}\in\Omega$. Note that we can write the differential equation (2.12) as a so-called 'gradient descent' equation:

$$\frac{d}{dt}J_i = -\frac{\partial}{\partial J_i}E[\boldsymbol{J}] \qquad E(\boldsymbol{J}) = \langle(\boldsymbol{J}\cdot\boldsymbol{x})\left[\theta(\boldsymbol{J}\cdot\boldsymbol{x}) - \theta(\boldsymbol{W}\cdot\boldsymbol{x})\right]\rangle_\Omega \tag{2.14}$$

  (which can be verified easily by explicit derivation). As a result we know that during the process (2.12) the quantity E can only decrease:

$$\frac{d}{dt}E[\boldsymbol{J}] = \sum_i \frac{\partial E}{\partial J_i}\frac{d}{dt}J_i = -\sum_i\left\{\frac{\partial E}{\partial J_i}\right\}^2 \leq 0$$

- In fact we can establish a positive lower bound for $|\frac{d}{dt}E[\boldsymbol{J}]|$ which holds for as long as $E > 0$, using the general inequality $\boldsymbol{y}^2 \geq (\boldsymbol{W}\cdot\boldsymbol{y})^2$ (this immediately follows from the Schwarz inequality, in combination with $|\boldsymbol{W}| = 1$):

$$\begin{aligned}\frac{d}{dt}E[\boldsymbol{J}] &= -\left[\frac{d}{dt}\boldsymbol{J}\right]^2 = -\langle\boldsymbol{x}[\theta(\boldsymbol{W}\cdot\boldsymbol{x}) - \theta(\boldsymbol{J}\cdot\boldsymbol{x})]\rangle^2 \\ &\leq -\langle(\boldsymbol{W}\cdot\boldsymbol{x})[\theta(\boldsymbol{W}\cdot\boldsymbol{x}) - \theta(\boldsymbol{J}\cdot\boldsymbol{x})]\rangle^2 \\ &= -\left\{\sum_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x})|\boldsymbol{W}\cdot\boldsymbol{x}|\theta\left[-(\boldsymbol{J}\cdot\boldsymbol{x})(\boldsymbol{W}\cdot\boldsymbol{x})\right]\right\}^2 \\ &\leq -\left\{\min_{\boldsymbol{x}\in\Omega^*(\boldsymbol{J})} p(\boldsymbol{x})|\boldsymbol{W}\cdot\boldsymbol{x}|\right\}^2\end{aligned}$$

  in which $\Omega^*(\boldsymbol{J}) = \{\boldsymbol{x}\in\Omega|\ (\boldsymbol{J}\cdot\boldsymbol{x})(\boldsymbol{W}\cdot\boldsymbol{x}) < 0\}$. This set is not empty if $E > 0$. We now immediately arrive at

$$E[\boldsymbol{J}] > 0: \qquad \frac{d}{dt}E[\boldsymbol{J}] \leq -\left\{\min_{\boldsymbol{x}\in\Omega} p(\boldsymbol{x})|\boldsymbol{W}\cdot\boldsymbol{x}|\right\}^2 = -K^2 \tag{2.15}$$

[1] A similar proof can be set up for the case where the input set is not discrete; here one will needs some additional assumptions such as $(\exists\delta > 0) : |\boldsymbol{W}\cdot\boldsymbol{x}| > \delta|\boldsymbol{x}|$ for all $\boldsymbol{x}\in\Omega$.

- Apparently the error $E(t) = E[\boldsymbol{J}(t)]$ obeys

$$E(t) \leq 0 \qquad \text{for} \qquad t > E(0)/K^2$$

On the other hand, we can prove that E is bounded from below (using $\theta(x) = \frac{1}{2}[\,\mathrm{sgn}(x) + 1]$):

$$E(\boldsymbol{J}) = \frac{1}{2}\langle(\boldsymbol{J}\cdot\boldsymbol{x})\,[\,\mathrm{sgn}(\boldsymbol{J}\cdot\boldsymbol{x}) - \,\mathrm{sgn}(\boldsymbol{W}\cdot\boldsymbol{x})]\rangle_\Omega$$

$$= \frac{1}{2}\langle|\boldsymbol{J}\cdot\boldsymbol{x}|\,[1 - \,\mathrm{sgn}(\boldsymbol{J}\cdot\boldsymbol{x})\,\mathrm{sgn}(\boldsymbol{W}\cdot\boldsymbol{x})]\rangle_\Omega \geq 0$$

As a consequence we know that $E(t) = 0$ for $t \geq E(0)/K^2$, which, in turn, implies that $\frac{d}{dt}J_i = 0\ \forall i$ as soon as $t > E(0)/K^2$. Therefore the vector $\boldsymbol{J}(t)$ evolves in a finite time towards a fixed-point. [2]

- Finally we show that any fixed-point will have the stated property. A fixed-point $\boldsymbol{J}$ satisfies

$$\langle\boldsymbol{x}\,[\,\mathrm{sgn}(\boldsymbol{W}\cdot\boldsymbol{x}) - \,\mathrm{sgn}(\boldsymbol{J}\cdot\boldsymbol{x})]\rangle_\Omega = 0$$

Taking the inner product with the 'teacher' vector $\boldsymbol{W}$ gives:

$$\langle(\boldsymbol{W}\cdot\boldsymbol{x})\,[\,\mathrm{sgn}(\boldsymbol{W}\cdot\boldsymbol{x}) - \,\mathrm{sgn}(\boldsymbol{J}\cdot\boldsymbol{x})]\rangle_\Omega = 0$$

$$\langle|\boldsymbol{W}\cdot\boldsymbol{x}|\,[1 - \,\mathrm{sgn}(\boldsymbol{W}\cdot\boldsymbol{x})\,\mathrm{sgn}(\boldsymbol{J}\cdot\boldsymbol{x})]\rangle_\Omega = 0$$

so, since $|\boldsymbol{W}\cdot\boldsymbol{x}| > 0\ \forall\boldsymbol{x}\in\Omega$:

$$\forall\boldsymbol{x}\in\Omega: \quad \mathrm{sgn}(\boldsymbol{W}\cdot\boldsymbol{x}) = \,\mathrm{sgn}(\boldsymbol{J}\cdot\boldsymbol{x})$$

which is equivalent to saying $M(\boldsymbol{x}) = \theta(\boldsymbol{J}\cdot\boldsymbol{x})\ \forall\boldsymbol{x}\in\Omega$. This completes our proof.

The quantity $E(\boldsymbol{J})$ (2.14) that is found to be minimised by the process (2.12) has a clear interpretation. By using the identity $\frac{1}{2}[1 - \,\mathrm{sgn}(A)\,\mathrm{sgn}(B)] = \theta[-AB]$, we can write $E(\boldsymbol{J})$ as

$$E(\boldsymbol{J}) = \langle|\boldsymbol{J}\cdot\boldsymbol{x}|\theta\,[-(\boldsymbol{J}\cdot\boldsymbol{x})(\boldsymbol{W}\cdot\boldsymbol{x})]\rangle_\Omega \tag{2.16}$$

Apparently $E(\boldsymbol{J})$ measures the average distance to the separating plane $\boldsymbol{J}\cdot\boldsymbol{x} = 0$ of all those input vectors $\boldsymbol{x}\in\Omega$ that are at the wrong side of this plane. Minimising E is therefore equivalent to moving the plane in such a way that this average distance goes down. At the minimum of E indeed the number of vectors at the wrong side of the plane is zero.

This interpretation of the learning rule as a gradient descent minimisation of some error measure (at least in the continuous time limit (2.12)) is an important one. Firstly, it allows us to build systematically other learning rules for our perceptron, by specifying alternative error measures, deriving the corresponding gradient descent equation, and subsequently performing the translation from the continuous time formulation back to the original discrete time formulation. Secondly, this interpretation also allows us to derive learning rules for the more complicated layered networks.

[2] A function E of a dynamical variable with these properties, bounded from below and descreasing monotonically with time, is called a Lyapunov function.

2.4 Learning in Layered Networks: Error Backpropagation

For feed-forward layered networks, where the state of any neuron cannot feed back to influence its input, the difference between graded response neurons and McCulloch-Pitts neurons is not too large. Consider two subsequent layers of graded response neurons, $\{U_i\}$ $(i = 1, \ldots, K)$ and $\{U'_j\}$ $(j = 1, \ldots, L)$, respectively. If the states in the first layer $\{U_i\}$ are stationary, the equations for the second layer are solved easily:

$$\tau \frac{d}{dt} U'_i(t) = \sum_k J_{ik} g[U_k - U^\star_k] - U'_i(t)$$

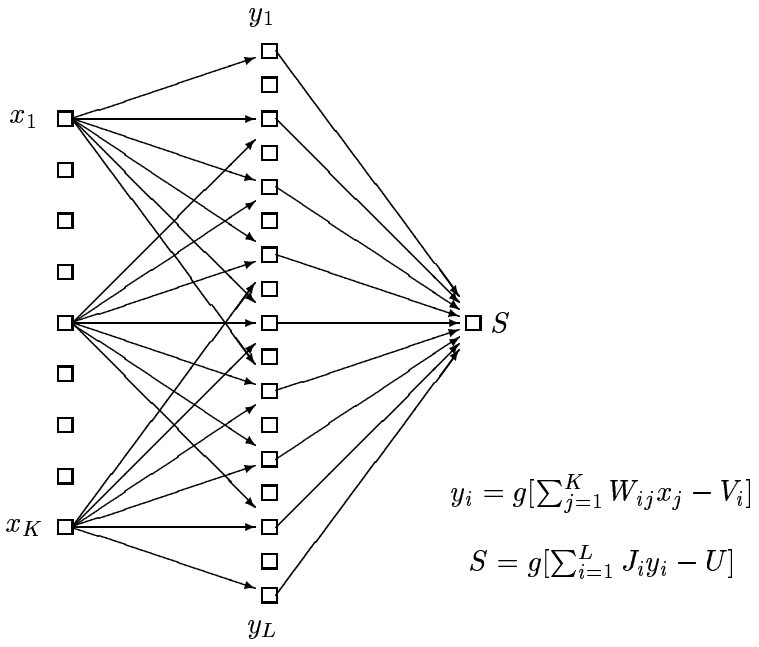
$$U'_i(t) = U'_i(0) e^{-t/\tau} + \sum_k J_{ik} g[U_k - U^\star_k] \left[1 - e^{-t/\tau}\right], \qquad U'_i(\infty) = \sum_k J_{ik} g[U_k - U^\star_k]$$

Therefore, if we simply wait until all layers have relaxed towards their stationary state $\{U'_i(\infty)\}$, one layer after the other, then for both McCulloch-Pitts neurons and graded response neurons the effective equations can eventually be written in the general form

$$S'_i = g\left[\sum_k J_{ik} S_k - U^\star_i\right] \tag{2.17}$$

with McCulloch-Pitts neurons just corresponding to the special choice $g[x] = \theta[x]$.

Single-Output Feed-Forward Two-Layer Networks. We now build a two-layer feedforward network of such units:



By using the familiar trick of adding the dummy variables $x_0 = y_0 = 1$ and defining $W_{i0} = -V_i$ and $J_0 = -U$, we can write our equations in their simpler form

$$y_i = g[\sum_{j=0}^{K} W_{ij} x_j] \qquad S = g[\sum_{i=0}^{L} J_i y_i] \tag{2.18}$$

or, in combination:

$$S(\boldsymbol{x}) = g\left[\sum_{i=0}^{L} J_i g[\sum_{j=0}^{K} W_{ij} x_j]\right] \tag{2.19}$$

The operation M to be learned need no longer be formulated in terms of binary variables, since the neurons produce real-valued output, so we write more generally:

$$M : \Omega \subseteq \Re^K \rightarrow [0,1] \tag{2.20}$$

Note, however, that the universality proof of two-layer feed-forward networks was given only for binary input- and output variables. The aim of a learning procedure is to achieve $S(\boldsymbol{x}) = M(\boldsymbol{x})\ \forall \boldsymbol{x} \in \Omega$. In the spirit of the situation we encountered with the continuous time version of the perceptron learning rule, we now define the dynamics of the weights by gradient descent on an error surface $E(\boldsymbol{W}, \boldsymbol{J})$ for which one usually chooses

$$E(\boldsymbol{W}, \boldsymbol{J}) = \frac{1}{2}\langle [S(\boldsymbol{x}) - M(\boldsymbol{x})]^2 \rangle_\Omega \tag{2.21}$$

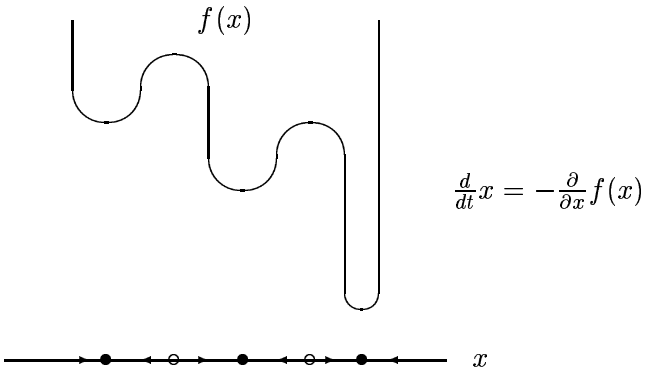
Clearly E is bounded from below; its minimum $E = 0$ corresponds to the desired situation where $S(\boldsymbol{x}) = M(\boldsymbol{x})\ \forall \boldsymbol{x} \in \Omega$. The learning rule thereby becomes:

$$\frac{d}{dt} W_{ij} = -\frac{\partial}{\partial W_{ij}} E(\boldsymbol{W}, \boldsymbol{J}) \qquad \frac{d}{dt} J_i = -\frac{\partial}{\partial J_i} E(\boldsymbol{W}, \boldsymbol{J}) \tag{2.22}$$

which guarantees

$$\frac{d}{dt} E = \sum_{ij} \frac{\partial E}{\partial W_{ij}} \frac{d}{dt} W_{ij} + \sum_i \frac{\partial E}{\partial J_i} \frac{d}{dt} J_i = -\sum_{ij} \left\{ \frac{\partial E}{\partial W_{ij}} \right\}^2 - \sum_i \left\{ \frac{\partial E}{\partial J_i} \right\}^2 \leq 0 \tag{2.23}$$

The overall error E will always go down, however, we have no guarantee that we will end up at the absolute minimum $E = 0$. A simple one-dimensional example will illustrate what could happen:



$$\frac{d}{dt} x = -\frac{\partial}{\partial x} f(x)$$

There are three local minima of $f(x)$ (indicated by $\bullet$), of which one is the global minimum, with two unstable fixed-points (indicated by $\circ$). Nevertheless, the dynamic equation $\frac{d}{dt} x = -\frac{\partial}{\partial x} f(x)$ will lead to the desired minimum only for initial values $x(0)$ in a restricted segment

of the x-axis. If $x(0)$ happens to be close to one of the other two local minima, the system will evolve towards a sub-optimal local minimum instead.

We can now simply use the chain rule for differentiation to work out the learning rules (2.22). We introduce the output error $\Delta(\boldsymbol{x}) = M(\boldsymbol{x}) - S(\boldsymbol{x})$ the network makes upon presentation of input vector $\boldsymbol{x} \in \Omega$, which allows us to write

$$\frac{d}{dt}J_i = \langle \frac{\partial S(\boldsymbol{x})}{\partial J_i}\Delta(\boldsymbol{x})\rangle_\Omega = \langle g'[\sum_{j=0}^{L} J_j y_j] y_i \Delta(\boldsymbol{x})\rangle_\Omega \tag{2.24}$$

$$\begin{aligned}\frac{d}{dt}W_{ij} &= \langle \frac{\partial S(\boldsymbol{x})}{\partial W_{ij}}\Delta(\boldsymbol{x})\rangle_\Omega = \langle g'[\sum_{k=0}^{L} J_k y_k] \sum_{\ell=0}^{L} J_\ell \frac{\partial y_\ell}{\partial W_{ij}}\Delta(\boldsymbol{x})\rangle_\Omega \\ &= \langle g'[\sum_{k=0}^{L} J_k y_k] \sum_{\ell=0}^{L} J_\ell \delta_{i\ell} g'[\sum_{m=0}^{K} W_{\ell m} x_m] x_j \Delta(\boldsymbol{x})\rangle_\Omega \\ &= \langle g'[\sum_{k=0}^{L} J_k y_k] J_i g'[\sum_{m=0}^{K} W_{im} x_m] x_j \Delta(\boldsymbol{x})\rangle_\Omega \end{aligned} \tag{2.25}$$

Usually the choice made for the nonlinear function $g[x]$ is something like $g[x] = \frac{1}{2}[1+\tanh(x)]$ or $g[x] = \mathrm{erf}(x)$, both of which have the property that the derivative of $g[x]$ will be some simple function of $g[x]$ itself, so $g'[x] = G[g[x]]$, in which case the learning rule simplifies to:

$$\frac{d}{dt}J_i = \langle G[S(\boldsymbol{x})] y_i \Delta(\boldsymbol{x})\rangle_\Omega \tag{2.26}$$

$$\frac{d}{dt}W_{ij} = \langle G[S(\boldsymbol{x})] J_i G[y_i(\boldsymbol{x})] x_j \Delta(\boldsymbol{x})\rangle_\Omega \tag{2.27}$$

We can now run backwards the derivation of the continuous time equation from the discrete modification rule, as discussed in the case of a perceptron, and turn this so-called batch-learning (2.26,2.27) into an on-line learning process:

$$J_i(t+\epsilon) = J_i(t) + \epsilon G[S(\boldsymbol{x})] y_i \Delta(\boldsymbol{x}) \tag{2.28}$$

$$W_{ij}(t+\epsilon) = W_{ij}(t) + \epsilon G[S(\boldsymbol{x})] J_i G[y_i(\boldsymbol{x})] x_j \Delta(\boldsymbol{x}) \tag{2.29}$$

$$\Updownarrow \quad (\epsilon \to 0)$$

$$\frac{d}{dt}J_i = \langle G[S(\boldsymbol{x})] y_i \Delta(\boldsymbol{x})\rangle_\Omega \tag{2.30}$$

$$\frac{d}{dt}W_{ij} = \langle G[S(\boldsymbol{x})] J_i G[y_i(\boldsymbol{x})] x_j \Delta(\boldsymbol{x})\rangle_\Omega \tag{2.31}$$

In the on-line equations (2.28,2.29), the vectors $\boldsymbol{x} \in \Omega$ are drawn at random at each iteration step. This discrete formulation, with the understanding that $\epsilon \ll 1$, is called 'learning by error backpropagation'. The term stemming from the property that, for the on-line formulation (2.28,2.29), by performing the chain rule differentiations, we are essentially linking all

variations in parameters appearing in an earlier layer of neurons to the resulting changes in the error $\Delta(\boldsymbol{x})$ in the outside layer.

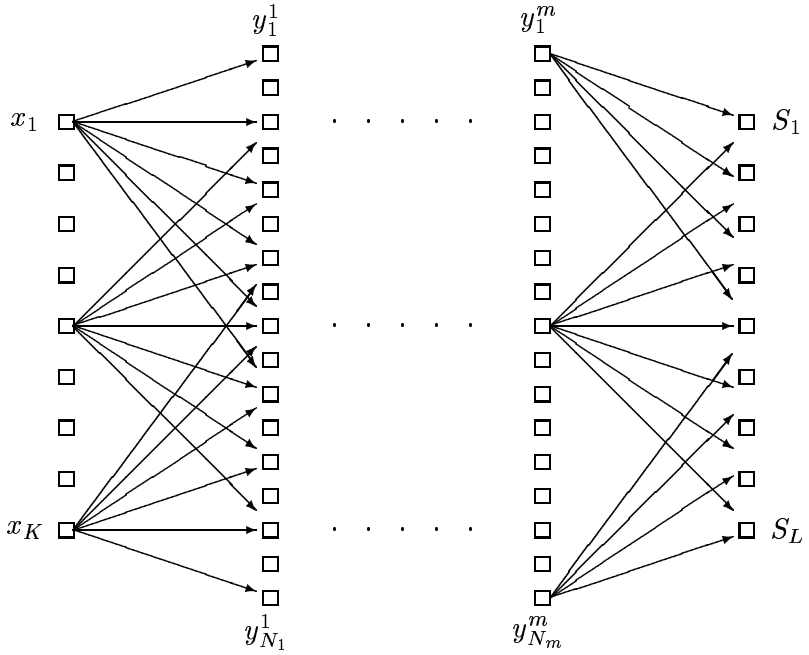
Generalisation to Arbitrary Feed-Forward Networks. The above construction can be generalised easily to an arbitrary number of output variables and an arbitrary number of hidden layers, as long as the structure remains of a strictly feed-forward nature.

We imagine a structure with K input variables and L output neurons. In between are m hidden layers, labelled by $\ell = 1, \ldots, m$, each of which containing N_ℓ neurons, the states of which are written as y_i^ℓ $(i = 1, \ldots, N_\ell)$. The weight connecting neuron j in layer $\ell - 1$ to neuron i in layer ℓ is denoted by J_{ij}^ℓ. The equations giving the neuron states then become (with the convention of the extra dummy neurons to take care of thresholds):

$$M : \Omega \subseteq \Re^K \to [0,1]^L \qquad S : \Omega \subseteq \Re^K \to [0,1]^L \tag{2.32}$$

$$y_i^1 = g\left[\sum_{j=0}^{K} J_{ij}^1 x_j\right], \qquad y_i^\ell = g\left[\sum_{j=0}^{N_{\ell-1}} J_{ij}^\ell y_j^{\ell-1}\right] \quad (\ell > 1) \tag{2.33}$$

$$S_i = g\left[\sum_{j=0}^{N_m} J_{ij}^{m+1} y_j^m\right] \tag{2.34}$$



The overall error measure is now defined as the sum of the squared errors of the L individual output neurons:

$$E(\boldsymbol{J}) = \frac{1}{2}\sum_{i=1}^{L}\langle[S_i(\boldsymbol{x}) - M_i(\boldsymbol{x})]^2\rangle_\Omega \tag{2.35}$$

E is bounded from below; its minimum $E = 0$ corresponds to the desired situation where $S_i(\boldsymbol{x}) = M_i(\boldsymbol{x})\ \forall \boldsymbol{x} \in \Omega$ and $\forall i \in \{1, \ldots, L\}$. The gradient descent learning rule generalises to:

$$\frac{d}{dt} J_{ij}^{\ell} = -\frac{\partial}{\partial J_{ij}^{\ell}} E(\boldsymbol{J}) \qquad \forall (i, j, \ell) \tag{2.36}$$

which guarantees

$$\frac{d}{dt} E = \sum_{ij} \sum_{\ell} \frac{\partial E}{\partial J_{ij}^{\ell}} \frac{d}{dt} J_{ij}^{\ell} = -\sum_{ij} \sum_{\ell} \left\{ \frac{\partial E}{\partial J_{ij}^{\ell}} \right\}^2 \leq 0 \tag{2.37}$$

To work out the learning rules (2.36), we again define output errors $\Delta_i(\boldsymbol{x}) = M_i(\boldsymbol{x}) - S_i(\boldsymbol{x})$ (of which there are L). We also simplify our equations by writing $g'[x] = G[g[x]]$. This allows us to write for $\ell = m + 1$:

$$\frac{d}{dt} J_{ij}^{m+1} = \sum_{k=1}^{L} \langle \frac{\partial S_k}{\partial J_{ij}^{m+1}} \Delta_k \rangle_{\Omega} = \langle \Delta_i G[S_i] y_j^m \rangle_{\Omega} \tag{2.38}$$

for $1 < \ell \leq m$:

$$\frac{d}{dt} J_{ij}^{\ell} = \sum_{k=1}^{L} \langle \frac{\partial S_k}{\partial J_{ij}^{\ell}} \Delta_k \rangle_{\Omega}$$

$$= \sum_{k=1}^{L} \sum_{i_\ell=0}^{N_\ell} \cdots \sum_{i_m=0}^{N_m} \langle \Delta_k \frac{\partial S_k}{\partial y_{i_m}^m} \frac{\partial y_{i_m}^m}{\partial y_{i_{m-1}}^{m-1}} \cdots \frac{\partial y_{i_{\ell+1}}^{\ell+1}}{\partial y_{i_\ell}^{\ell}} \frac{\partial y_{i_\ell}^{\ell}}{\partial J_{ij}^{\ell}} \rangle_{\Omega}$$

$$= \sum_{k=1}^{L} \sum_{i_\ell=0}^{N_\ell} \cdots \sum_{i_m=0}^{N_m} \langle \Delta_k G[S_k] J_{k i_m}^{m+1} G[y_{i_m}^m] J_{i_m i_{m-1}}^m \cdots G[y_{i_{\ell+1}}^{\ell+1}] J_{i_{\ell+1} i_\ell}^{\ell+1} G[y_{i_\ell}^{\ell}] \delta_{i, i_\ell} y_j^{\ell-1} \rangle_{\Omega}$$

$$= \sum_{k=1}^{L} \sum_{i_{\ell+1}=0}^{N_{\ell+1}} \cdots \sum_{i_m=0}^{N_m} \langle \Delta_k G[S_k] J_{k i_m}^{m+1} G[y_{i_m}^m] J_{i_m i_{m-1}}^m \cdots G[y_{i_{\ell+1}}^{\ell+1}] J_{i_{\ell+1} i_\ell}^{\ell+1} G[y_i^{\ell}] y_j^{\ell-1} \rangle_{\Omega} \tag{2.39}$$

and, finally, for $\ell = 1$ we only have to change the last derivative in the previous chain:

$$\frac{d}{dt} J_{ij}^1 = \sum_{k=1}^{L} \sum_{i_2=0}^{N_2} \cdots \sum_{i_m=0}^{N_m} \langle \Delta_k G[S_k] J_{k i_m}^{m+1} G[y_{i_m}^m] J_{i_m i_{m-1}}^m \cdots G[y_{i_2}^2] J_{i_2 i_1}^2 G[y_i^1] x_j \rangle_{\Omega} \tag{2.40}$$

The discrete-time on-line version of this procedure is obtained in the by now familiar manner. The average over the set Ω is removed and at each iteration step an input vector $\boldsymbol{x} \in \Omega$ is drawn at random instead:

$$J_{ij}^{m+1}(t + \epsilon) = J_{ij}^{m+1}(t) + \epsilon \Delta_i G[S_i] y_j^m \tag{2.41}$$

for $1 < \ell \leq m$:

$$J_{ij}^{\ell}(t+\epsilon) = J_{ij}^{\ell}(t) + \epsilon \sum_{k=1}^{L} \sum_{i_{\ell+1}=0}^{N_{\ell+1}} \cdots \sum_{i_m=0}^{N_m} \Delta_k G[S_k] J_{k i_m}^{m+1} G[y_{i_m}^m] J_{i_m i_{m-1}}^m \cdots G[y_{i_{\ell+1}}^{\ell+1}] J_{i_{\ell+1} i_\ell}^{\ell+1} G[y_i^{\ell}] y_j^{\ell-1} \tag{2.42}$$

and for $\ell = 1$:

$$J^1_{ij}(t+\epsilon) = J^1_{ij} + \epsilon \sum_{k=1}^{L} \sum_{i_2=0}^{N_2} \cdots \sum_{i_m=0}^{N_m} \Delta_k G[S_k] J^{m+1}_{ki_m} G[y^m_{i_m}] J^m_{i_m i_{m-1}} \cdots G[y^2_{i_2}] J^2_{i_2 i_1} G[y^1_i] x_j \quad (2.43)$$

with the learning rate $\epsilon \ll 1$. The above equations illustrate more clearly than in the two-layer case the suggestion of errors propagating backwards through the network to the parameter that is being modified.

2.5 Dynamics of Learning in Large Perceptrons

Here we show how for large perceptrons and simple probability distributions for the input vectors $\boldsymbol{x} \in \Omega$ the dynamical equation for the connections $\boldsymbol{J}(t)$ can be reduced to just two coupled non-linear differential equations, which contain all the relevant information on the learning process and the performance of the perceptron at any time. In particular, we will find the generalisation error of the perceptron as a function of time (i.e. given the desired reliability of the system's operation, we know the required duration of the learning stage).

Macroscopic Observables. Our starting point is the Ising perceptron described by the continuous time equation (2.13), with the linearly separable task $T(\boldsymbol{x}) = \text{sgn}(\boldsymbol{W} \cdot \boldsymbol{x})$. By taking the inner product in both sides with $\boldsymbol{J}$ and $\boldsymbol{W}$ we obtain the following two equations:

$$\frac{d}{dt}\boldsymbol{J}^2 = 2\boldsymbol{J} \cdot \frac{d}{dt}\boldsymbol{J} = \langle (\boldsymbol{J} \cdot \boldsymbol{x}) \, [\, \text{sgn}(\boldsymbol{W} \cdot \boldsymbol{x}) - \text{sgn}(\boldsymbol{J} \cdot \boldsymbol{x})] \rangle_\Omega$$

$$\frac{d}{dt}(\boldsymbol{W} \cdot \boldsymbol{J}) = \frac{1}{2} \langle (\boldsymbol{W} \cdot \boldsymbol{x}) \, [\, \text{sgn}(\boldsymbol{W} \cdot \boldsymbol{x}) - \text{sgn}(\boldsymbol{J} \cdot \boldsymbol{x})] \rangle_\Omega$$

We now put $\boldsymbol{J} = J\hat{\boldsymbol{J}}$, with $|\hat{\boldsymbol{J}}| = 1$, and $\omega = \boldsymbol{W} \cdot \hat{\boldsymbol{J}}$. Using relations like $\frac{d}{dt}(\boldsymbol{W} \cdot \boldsymbol{J}) = J\frac{d}{dt}\omega + \omega\frac{d}{dt}J$ we obtain the following two equations in terms of the two macroscopic observables J and ω:

$$\frac{d}{dt}J = \frac{1}{2} \langle (\hat{\boldsymbol{J}} \cdot \boldsymbol{x}) \Big[\, \text{sgn}(\boldsymbol{W} \cdot \boldsymbol{x}) - \text{sgn}(\hat{\boldsymbol{J}} \cdot \boldsymbol{x}) \Big] \rangle_\Omega$$

$$J\frac{d}{dt}\omega = \frac{1}{2} \langle \Big[(\boldsymbol{W} \cdot \boldsymbol{x}) - \omega(\hat{\boldsymbol{J}} \cdot \boldsymbol{x}) \Big] \, \Big[\, \text{sgn}(\boldsymbol{W} \cdot \boldsymbol{x}) - \text{sgn}(\hat{\boldsymbol{J}} \cdot \boldsymbol{x}) \Big] \rangle_\Omega$$

Note that the statistics of the input vectors $\boldsymbol{x}$ enter into these equations only through the two quantities $u = \boldsymbol{W} \cdot \boldsymbol{x}$ and $v = \hat{\boldsymbol{J}} \cdot \boldsymbol{x}$, so that we can write

$$\frac{d}{dt}J = \frac{1}{2} \int du dv \; P(u,v) v \, [\, \text{sgn}(u) - \text{sgn}(v)]$$

$$J\frac{d}{dt}\omega = \frac{1}{2} \int du dv \; P(u,v) \, [u - \omega v] \, [\, \text{sgn}(u) - \text{sgn}(v)]$$

in which $P(u,v)$ denotes the joint probability distribution for the variables $u = \boldsymbol{W} \cdot \boldsymbol{x}$ and $v = \hat{\boldsymbol{J}} \cdot \boldsymbol{x}$, where $\boldsymbol{x} \in \Omega$ with probability $p(\boldsymbol{x})$. This distribution depends on time through the evolving vector $\hat{\boldsymbol{J}}$.

By working out the effect of the sgn() functions for the various integration regimes, the above differential equations can be written in the simple form

$$\frac{d}{dt}J = -\int_0^\infty\int_0^\infty dudv\ v\,[P(u,-v)+P(-u,v)] \tag{2.44}$$

$$\frac{d}{dt}\omega = \frac{1}{J}\int_0^\infty\int_0^\infty dudv\ \ [u+\omega v]\,[P(u,-v)+P(-u,v)] \tag{2.45}$$

Clearly $\frac{d}{dt}J \leq 0$ and $\frac{d}{dt}\omega \geq 0$ at all times. In order to quantify the degree to which the perceptron has learned the task at hand, and to interpret the dynamic equations (2.44,2.45), we can define and calculate some error measures. The quantity $E(\boldsymbol{J})$ (2.14), for instance, can be rewritten in terms of the above variables as

$$E = J\int_0^\infty\int_0^\infty dudv\ v\,[P(u,-v)+P(-u,v)] \tag{2.46}$$

Another popular quantity is the so-called *generalisation error* E_g, defined as the total probability of finding an input vector $\boldsymbol{x}\in\Omega$ such that $\mathrm{sgn}(\boldsymbol{W}\cdot\boldsymbol{x}) \neq \mathrm{sgn}(\boldsymbol{J}\cdot\boldsymbol{x})$. Its definition $E_g = \langle\theta[-(\boldsymbol{W}\cdot\boldsymbol{x})(\boldsymbol{J}\cdot\boldsymbol{x})]\rangle_\Omega$ translates immediately into

$$E_g = \int_0^\infty\int_0^\infty dudv\ \ [P(u,-v)+P(-u,v)] \tag{2.47}$$

All these equations are still completely general. The specific details of the task, of N and the distribution of input vectors are contained in $P(u,v)$. The length J of the evolving vector $\boldsymbol{J}$ always decreases monotonically. As soon as $J=0$, we will have to define in the original dynamical rules what we want sgn(0) to be (the same, of course, occurs in the original perceptron learning rule); the usual and most natural choice is $\mathrm{sgn}(0)=0$.

Large Perceptrons with Uniformly Distributed Input Vectors. The three important effects of studying the limit of large perceptrons, $N\to\infty$, and uniformly distributed input vectors, $\Omega=\{-1,1\}^N$ with $p(\boldsymbol{x})=2^{-N}\ \forall\boldsymbol{x}$, are (i) that we can calculate $P(u,v)$, (ii) that $P(u,v)$ turns out to be a rather simple function, and (iii) that $P(u,v)$ depends on time only through ω. This means that the two equations (2.44,2.45) close, i.e. their right-hand sides can be expressed solely as functions of (ω,J). Instead of N coupled nonlinear differential equations (for the N components of the wheight vector $\boldsymbol{J}$, where $N\to\infty$) we have reduced the problem to just two coupled nonlinear differential equations, which in fact can subsequently be further reduced to only one. Not all of the properties assumed here to arrive at this simplification are, strictly speaking, necessary. In fact we only need $N\to\infty$ and $p(\boldsymbol{x})=\prod_i p_i(x_i)$ to play the game, and we can even apply the same ideas to multilayer networks, but here we will just illustrate the simplest case.

For $N\to\infty$ and $\Omega=\{-1,1\}^N$ with $p(\boldsymbol{x})=2^{-N}\ \forall\boldsymbol{x}\in\Omega$ (so that all components of $\boldsymbol{x}$ are distributed independently according to $p(x_i)=\frac{1}{2}$ for $x_i=\pm1$), we can try to apply the central limit theorem to the two stochastic quantities $u=\boldsymbol{W}\cdot\boldsymbol{x}$ and $v=\hat{\boldsymbol{J}}\cdot\boldsymbol{x}$, which would suggest that they are described by a Gaussian probability distribution $P(u,v)$. This will usually be true, but not always (one simple counterexample is the case where $\boldsymbol{W}$ or $\hat{\boldsymbol{J}}$ has only a *finite* number of nonzero components). The precise condition for an inner product $\boldsymbol{W}\cdot\boldsymbol{x}$ to acquire a Gaussian probability distribution for $N\to\infty$, with statistically

independent random components $x_i \in \{-1, 1\}$ (with equal probabilities), is rather subtle (see e.g. W. Feller, 1966, 'An Introduction to Probability and its Applications II', Wiley, New York). A sufficient condition is

$$\forall \epsilon > 0: \qquad \lim_{N\to\infty} \sum_{i=1}^{N} \theta\left[W_i^2 - \epsilon \sum_{k=1}^{N} W_k^2\right] = 0 \tag{2.48}$$

A necessary condition is

$$\lim_{N\to\infty} \frac{\sum_{i=1}^{N} W_i^4}{[\sum_{i=1}^{N} W_i^2]^2} = 0 \tag{2.49}$$

(for a derivation of these conditions see Appendix A). Both conditions state that for large N the inner product $\boldsymbol{W}\cdot\boldsymbol{x}$ should not be dominated by just a small number of components of $\boldsymbol{W}$.

In the generic case both conditions (2.48,2.49) are fulfilled , and $P(u,v)$ is a Gaussian distribution. In appendix B we derive some general results for Gaussian distributions of more than one variable. For the present problem we can calculate directly the first few moments of $P(u,v)$ (note: $\langle x_i\rangle_\Omega = 0$, $\langle x_i x_j\rangle_\Omega = \delta_{ij}$):

$$\begin{array}{llll}
\int dudv\ u\ P(u,v) & = \langle u\rangle_\Omega & = \sum_{i=1}^{N} W_i\langle x_i\rangle_\Omega & = 0 \\
\int dudv\ v\ P(u,v) & = \langle v\rangle_\Omega & = \sum_{i=1}^{N} \hat{J}_i\langle x_i\rangle_\Omega & = 0 \\
\int dudv\ u^2 P(u,v) & = \langle u^2\rangle_\Omega & = \sum_{ij=1}^{N} W_i W_j\langle x_i x_j\rangle_\Omega & = \sum_{i=1}^{N} W_i^2 = 1 \\
\int dudv\ v^2 P(u,v) & = \langle v^2\rangle_\Omega & = \sum_{ij=1}^{N} \hat{J}_i \hat{J}_j\langle x_i x_j\rangle_\Omega & = \sum_{i=1}^{N} \hat{J}_i^2 = 1 \\
\int dudv\ uv\ P(u,v) & = \langle uv\rangle_\Omega & = \sum_{ij=1}^{N} W_i \hat{J}_j\langle x_i x_j\rangle_\Omega & = \sum_{i=1}^{N} W_i \hat{J}_i = \omega
\end{array} \tag{2.50}$$

Using the results of Appendix A, we know that the values of the moments (2.50) completely determine the distribution $P(u,v)$:

$$P(u,v) = \frac{\sqrt{\det \boldsymbol{A}}}{2\pi}\ e^{-\frac{1}{2}\begin{pmatrix} u \\ v \end{pmatrix}\cdot \boldsymbol{A} \begin{pmatrix} u \\ v \end{pmatrix}} \qquad \boldsymbol{A}^{-1} = \begin{pmatrix} \langle u^2\rangle & \langle uv\rangle \\ \langle uv\rangle & \langle v^2\rangle \end{pmatrix} = \begin{pmatrix} 1 & \omega \\ \omega & 1 \end{pmatrix}$$

All we need to do is to invert the matrix $\boldsymbol{A}^{-1}$, which gives the desired result:

$$\boldsymbol{A} = \frac{1}{1-\omega^2}\begin{pmatrix} 1 & -\omega \\ -\omega & 1 \end{pmatrix} \qquad \det \boldsymbol{A} = \frac{1}{1-\omega^2}$$

$$P(u,v) = \frac{1}{2\pi\sqrt{1-\omega^2}}\ e^{-\frac{1}{2}[u^2+v^2-2\omega uv]/(1-\omega^2)} \tag{2.51}$$

Without having to calculate integrals, we can already draw important conclusions on the solution of the differential equations (2.44,2.45), by simply exploiting the property that (2.51) is invariant under the permutation $u \leftrightarrow v$, leading to the identities $P(u,-v) = P(-u,v)$ and

$$\int_0^\infty\int_0^\infty dudv\ v\,[P(u,-v)+P(-u,v)] = \int_0^\infty\int_0^\infty dudv\ u\,[P(u,-v)+P(-u,v)]$$

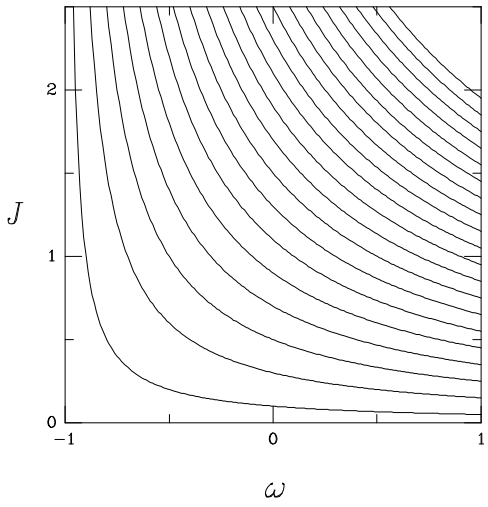


Figure 2.1: The relation between $J(t)$ and $\omega(t)$ during the training of large perceptrons with small learning rates, on linearly separable tasks. Different curves refer to different initial conditions $(\omega(0), J(0))$. The flow through these trajectories is from the left to the right.

As a result we can simplify (2.44,2.45) to

$$\frac{d}{dt}J = -K(\omega) \qquad \frac{d}{dt}\omega = \frac{1+\omega}{J}K(\omega) \tag{2.52}$$

$$K(\omega) = \int_0^\infty\int_0^\infty \frac{du\,dv}{\pi\sqrt{1-\omega^2}}\; v\; e^{-\frac{1}{2}[u^2+v^2+2\omega uv]/(1-\omega^2)} \tag{2.53}$$

Elimination of $K(\omega)$ from the two equations (2.52) gives $(1+\omega)^{-1}\frac{d}{dt}\omega + J^{-1}\frac{d}{dt}J = 0$, or

$$\frac{d}{dt}\Big\{\log(1+\omega)+\log(J)\Big\} = 0 \qquad \Rightarrow \qquad J(t) = \frac{C}{1+\omega(t)} \tag{2.54}$$

in which the constant can be determined by inserting the details of the initial state: $C = J(0)[1+\omega(0)]$. The curves (2.54), for various values of C, are shown in figure 2.1.

In order to go further, and calculate the values of the macroscopic observables as functions of time, we calculate in Appendix A the integral $K(\omega)$ (2.53). The result is

$$K(\omega) = \frac{1-\omega}{\sqrt{2\pi}}$$

with which we obtain for (2.52)

$$\frac{d}{dt}J = -\frac{1-\omega}{\sqrt{2\pi}} \qquad \frac{d}{dt}\omega = \frac{1-\omega^2}{J\sqrt{2\pi}} \tag{2.55}$$

Finally we can use the relation between J and ω (2.54) to reduce the set (2.55) to just a single equation for $\omega(t)$. We simply substitute $J(t) = J(0)[1+\omega(0)]/[1+\omega(t)]$ into the differential equation for $\omega(t)$:

$$\frac{d}{dt}\omega = \frac{1}{D}[1+\omega][1-\omega^2] \qquad D = J(0)[1+\omega(0)]\sqrt{2\pi} \tag{2.56}$$

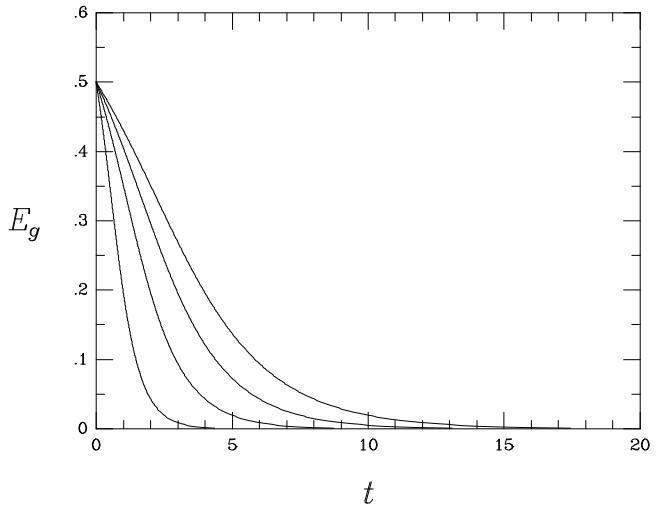


Figure 2.2: The generalisation error E_g as a function of time, following a random initial weight configuration, i.e. $E_g(0) = \frac{1}{2}$. The different curves refer to $J(0) = 2,\ \frac{3}{2},\ 1,\ \frac{1}{2}$ (from top to bottom).

This equation is somewhat nasty to solve. The inverse equation for $\frac{d}{d\omega}t$, in contrast, is solved easily:

$$\frac{d}{d\omega}t = \frac{D}{[1+\omega][1-\omega^2]} = \frac{D}{2[1+\omega]}\left[\frac{1}{1+\omega} + \frac{1}{1-\omega}\right]$$

$$= \frac{1}{2}D\left[\frac{1}{[1+\omega]^2} + \frac{1}{1-\omega^2}\right]$$

$$= \frac{1}{4}D\left[\frac{2}{[1+\omega]^2} + \frac{1}{1+\omega} + \frac{1}{1-\omega}\right]$$

$$= \frac{1}{4}D\frac{d}{d\omega}\left\{-\frac{2}{1+\omega} + \log[1+\omega] - \log[1-\omega]\right\}$$

So that

$$t = J(0)[1+\omega(0)]\sqrt{\frac{\pi}{2}}\left\{\log\left[\frac{1+\omega}{1-\omega}\right]^{\frac{1}{2}} - \frac{1}{1+\omega} + A\right\} \tag{2.57}$$

with the constant A determined by initial conditions:

$$A = \frac{1}{1+\omega(0)} - \log\left[\frac{1+\omega(0)}{1-\omega(0)}\right]^{\frac{1}{2}} \tag{2.58}$$

In particular, for the simple and common case where the initial student vector $\boldsymbol{J}(0)$ is drawn at random, we typically have $\omega(0) = 0$. The solution (2.57) now takes a simple form:

$$\omega(0) = 0: \qquad t = J(0)\sqrt{\frac{\pi}{2}}\left\{\log\left[\frac{1+\omega}{1-\omega}\right]^{\frac{1}{2}} + \frac{\omega}{1+\omega}\right\} \tag{2.59}$$

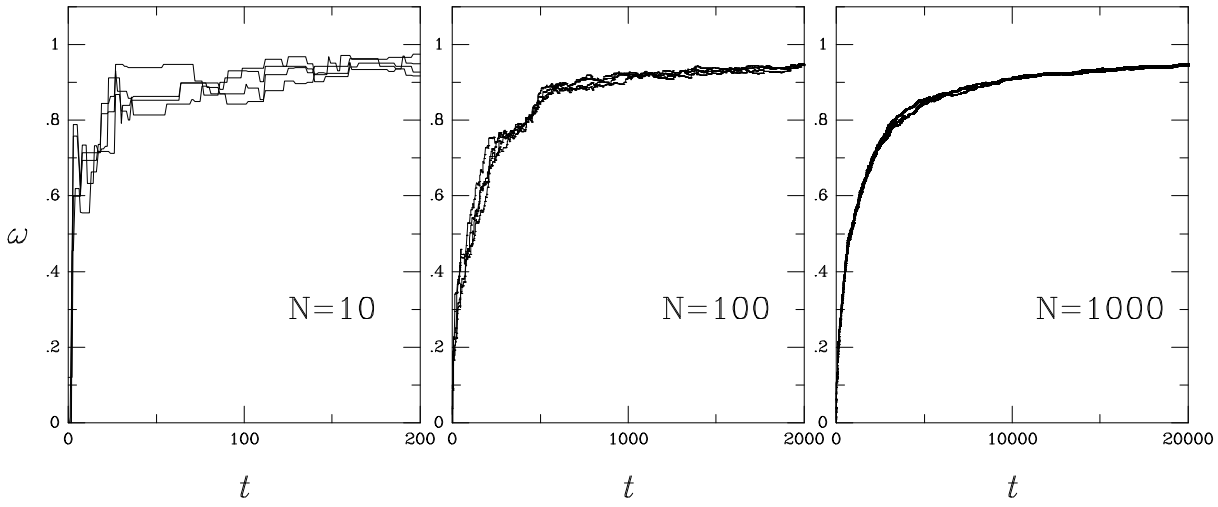


Figure 2.3: Evolution in time of the macroscopic observables $J = |\boldsymbol{J}|$ and $\omega = \hat{\boldsymbol{J}} \cdot \boldsymbol{W}$ for a numerical simulation of the standard discrete-time perceptron learning rule (with a randomly drawn task vector $\boldsymbol{W}$ and learning rate $\epsilon = 1$), initialised randomly. In each picture: $J(0) \in \{2, \frac{3}{2}, 1, \frac{1}{2}\}$.

For large perceptrons we can also work out the precise relation (2.47) between ω and the generalisation error E_g (2.47) (see Appendix A), which gives

$$E_g = \frac{1}{\pi}\arccos(\omega) \qquad \omega = \cos(\pi E_g) \tag{2.60}$$

This one-to-one relation enables us to write the equations for ω (2.56,2.57) in terms of E_g. For instance, by using standard relations like $\cos(2\alpha) = \cos^2(\alpha) - \sin^2(\alpha)$, we can transform the result (2.59) (referring to $\omega(0) = 0$, $E_g(0) = \frac{1}{2}$) into:

$$t = \frac{1}{2}\sqrt{\frac{\pi}{2}}J(0)\left\{1 - \tan^2(\frac{1}{2}\pi E_g) - 2\log\tan(\frac{1}{2}\pi Eg)\right\} \tag{2.61}$$

(drawn in figure 2.2).

2.6 Numerical Simulations

We end this chapter with some numerical simulations of the various learning procedures and architectures. We will illustrate some general trends and, were possible, make comparisons with theoretical results.

Numerical Simulations of Perceptrons. We first simulate numerically the original discrete perceptron learning procedure (2.4), for different choices of the input space dimension N and the initial length $J(0) = |\boldsymbol{J}(0)|$ of the student vector. For simplicity we choose $\Omega = \{-1, 1\}^N$, $p(\boldsymbol{x}) = 2^{-N}$ $\forall \boldsymbol{x} \in \Omega$, and a randomly drawn teacher vector $\boldsymbol{W}$. The results are shown in figure 2.3. Several conclusions can be drawn from such experiments (keeping in mind that for specifically constructed pathological teacher vectors the picture might be different):

1. The duration of the learning process scales *linearly* with N.

2. If viewed on the relevant N-dependent time-scale (as in the figure), the fluctuations in ω (due to the randomness in the selection of input vectors $\boldsymbol{x}$) become negligible as $N \to \infty$ (this is in fact the property which enabled the derivation of the deterministic continuous-time equation).

3. In contrast to the situation with small learning rates, for $\epsilon = 1$ all differences in the initial length $J(0)$ which are of order $\mathcal{O}(1)$ are irrelevant (clearly: they can be wiped out in just a few iteration steps).

Next we show how for $N \to \infty$ and $\epsilon \to 0$ the learning dynamics is indeed described by equation (2.57). We have to keep in mind that in the derivation we have taken the two limits in a specific order: $\lim_{\epsilon \to 0}$ followed by $\lim_{N \to \infty}$. The required ϵ therefore may depend on N: $\epsilon = \epsilon(N)$. Since ϵ defines the unit of time in the process $\boldsymbol{J}(t + \epsilon) = \boldsymbol{J}(t) + \epsilon \Delta \boldsymbol{J}(\ldots)$, and since the learning time is found to scale linearly with N, we are led to the scaling relation

$$\epsilon(N) = \tilde{\epsilon} N^{-1}, \qquad \tilde{\epsilon} \ll 1 \tag{2.62}$$

According to the simulation experiments for small ϵ, as shown in figures 2.4 and 2.5 this is indeed the appropriate scaling. Note that the time can here no longer be identified with the number of iteration steps (as in the $\epsilon = 1$ case), for $\epsilon < 1$ the relation is $t = n_{\rm steps}\epsilon$. Equivalently,

$$n_{\rm steps} = Nt/\tilde{\epsilon}$$

Note also that the roles played by the two limits $N \to \infty$ and $\tilde{\epsilon} \to 0$ are different. Decreasing $\tilde{\epsilon}$ moves the experimental curves towards the theoretical ones; increasing N reduces the fluctuations (mainly in initial conditions).

Numerical Simulations of Error Backpropagation. In the case of multi-layer networks, trained with the on-line version of error backpropagation (i.e. the version where at each iteration step an input vector is drawn at random), there is little theory to compare with (apart from results on the so-called 'committee machines'; two-layer networks in which only the weights feeding into the hidden layer evolve in time). Therefore we will just show how the procedure works out in practice, by measuring as a function of time, for a network with two layers and a single output neuron, the output error $E = \frac{1}{2}\langle[S(\boldsymbol{x}) - M(\boldsymbol{x})]^2\rangle_\Omega$ for two simple tasks: the parity operation (in ± 1 representation) and a linearly separable operation (with a randomly drawn teacher vector):

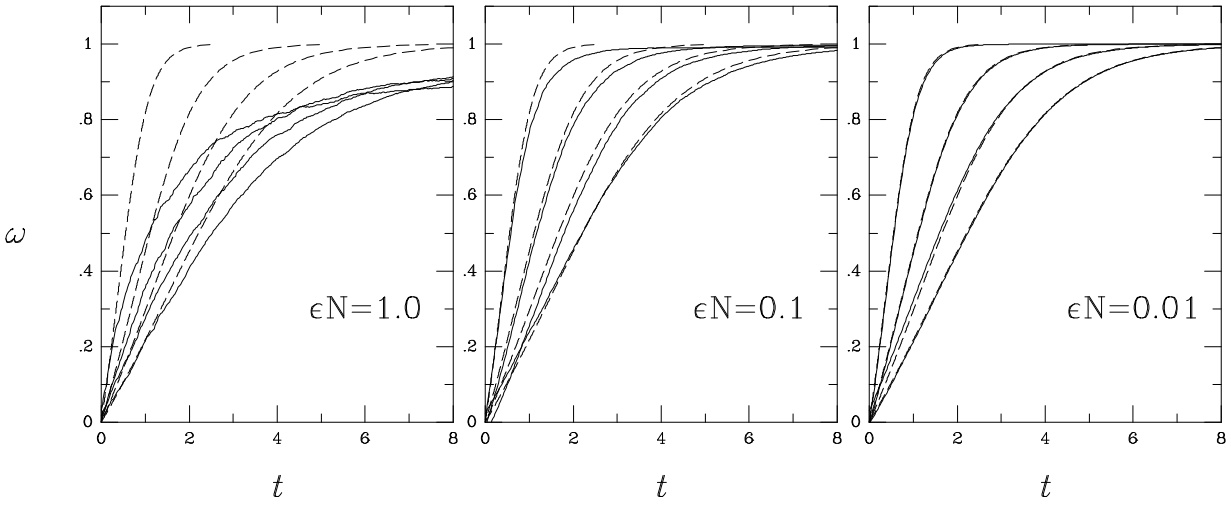


Figure 2.4: Evolution of ω in an Ising perceptron (with $N = 1000$ and a randomly drawn task vector $\boldsymbol{W}$), for $\epsilon N \in \{1,\ 0.1,\ 0.01\}$, following random initialisation. In each picture: Solid lines = numerical simulations; dashed lines = theory ($J(0) \in \{2, \frac{3}{2}, 1, \frac{1}{2}\}$, from top to bottom).

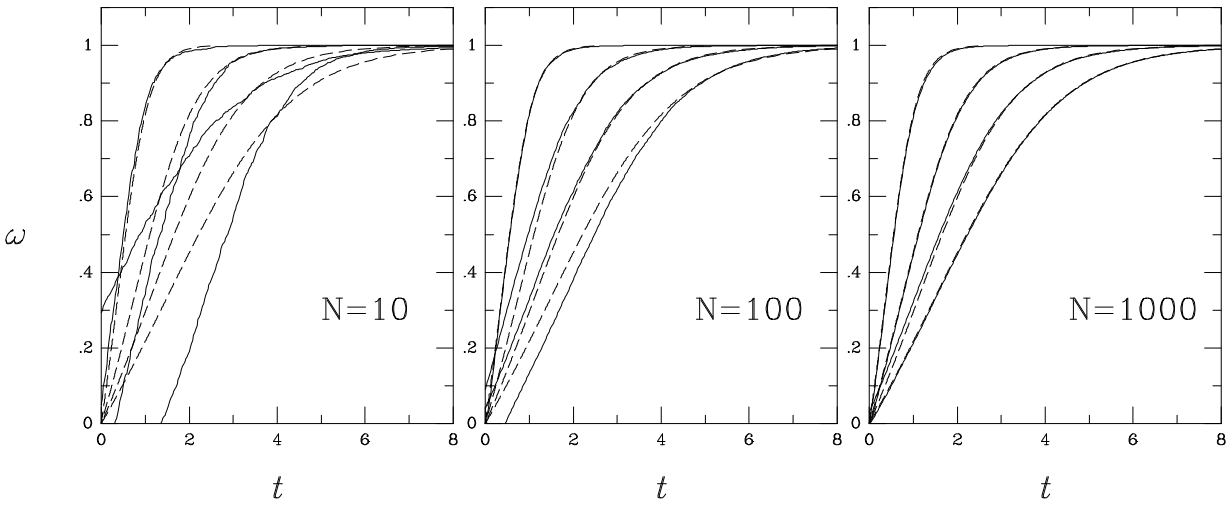


Figure 2.5: Evolution of ω in an Ising perceptron (with $\epsilon = 0.01/N$ and a randomly drawn task vector $\boldsymbol{W}$), for $N \in \{10,\ 100,\ 1000\}$, following random initialisation. In each picture: Solid lines = numerical simulations; dashed lines = theory ($J(0) \in \{2, \frac{3}{2}, 1, \frac{1}{2}\}$, from top to bottom).

$$\boldsymbol{x} \in \Omega = \{-1,1\}^K \qquad \begin{array}{ll} \text{task I}: & M(\boldsymbol{x}) = \prod_{i=1}^K x_i \in \{-1,1\} \\ \text{task II}: & M(\boldsymbol{x}) = \ \text{sgn}(\boldsymbol{W}\cdot\boldsymbol{x}) \in \{-1,1\} \end{array} \tag{2.63}$$

to be learned by the two-layer network

$$S(\boldsymbol{x}) = g\left[\sum_{i=0}^{L} J_i y_i(\boldsymbol{x})\right] \qquad y_i(\boldsymbol{x}) = g\left[\sum_{j=0}^{K} W_{ij} x_j\right] \qquad g[x] = \tanh(x) \in \langle -1,1\rangle \tag{2.64}$$

with the usual dummy variables $x_0 = y_0 = 1$. Perfect performance would correspond to $E = 0$. On the other hand, for a task $M(\boldsymbol{x}) \in \{-1,1\}$ a trivial perceptron with zero parameters (weights and thresholds) throughout, would give $E = \frac{1}{2}\langle M^2(\boldsymbol{x})\rangle_\Omega = \frac{1}{2}$.

The results are shown in figures 2.6, 2.7 and 2.8 (all these results involve four independent trials for each parameter combination and each task). For the on-line version of error backpropagation to approach the learning equations involving *averages* over the input set (the 'batch' version, corresponding to gradient descent on the error surface), we again have to take the limit $\epsilon \to 0$ for every (K,L) combination, so we should expect $\epsilon = \epsilon(K,L)$. The simulation results indicate

$$\epsilon(K,L) = \tilde{\epsilon} K^{-1}, \qquad \tilde{\epsilon} \ll 1 \tag{2.65}$$

We have to be careful, however, in drawing conclusions about whether the network succeeds in solving the problem from experiments such as the ones in figures 2.6, 2.7 and 2.8. A learning procedure can involve several distinct stages and time-scales (possibly dependent on initial conditions). For instance, the experiments done in the time window $0 < t < 100$ suggest that the network does not succeed in performing the parity operation for $(K,L) \in \{(10,10),(10,15)\}$ (whereas we know it is capable of doing so for $L \geq K$). However, if we enlarge our time window we do find a certain fraction of our trials being succesful (one just has to wait longer), as is illustrated by figure 2.9. The system apparently spends a significant amount of time in a transient so-called 'plateau' phase, where the error E does not change much, before it discovers the rule underlying the training examples.

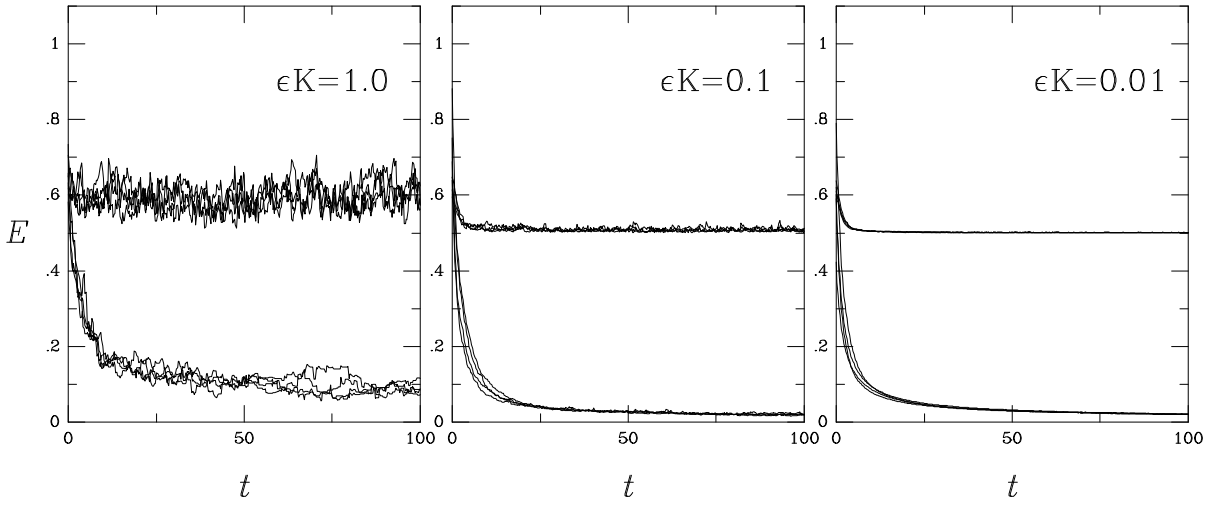


Figure 2.6: Evolution of the overall error E in a two-layer feed-forward network, trained by error backpropagation (with $K = 15$ input neurons, $L = 10$ hidden neurons, and a single output neuron). The results refer to independent experiments involving either a linearly separable task (with random teacher vector, lower curves) or the parity operation (upper curves), with $\epsilon K \in \{1,\ 0.1,\ 0.01\}$, following random initialisation.

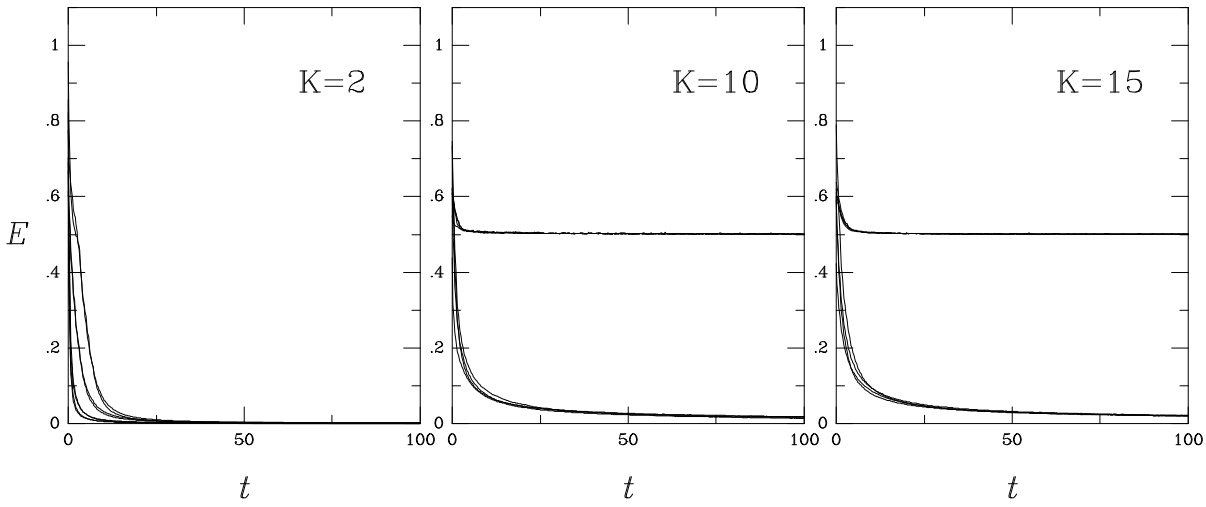


Figure 2.7: Evolution of the overall error E in a two-layer feed-forward network, trained by error backpropagation (with $\epsilon K = 0.01$, $L = 10$ hidden neurons, and a single output neuron). The results refer to independent experiments involving either a linearly separable task (with random teacher vector, lower curves) or the parity operation (upper curves), with $K \in \{2,\ 10,\ 15\}$, following random initialisation.

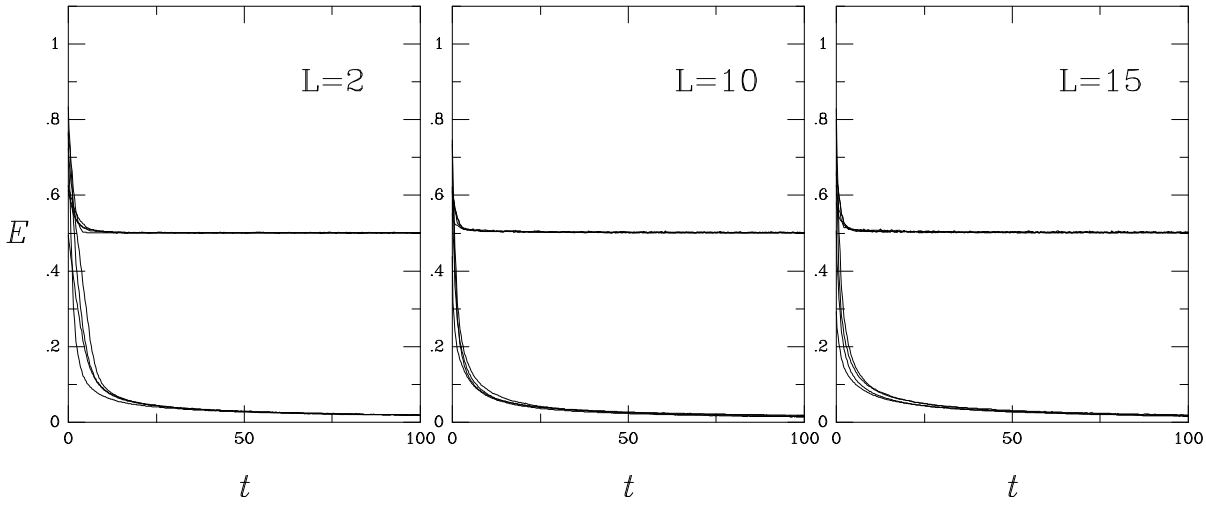


Figure 2.8: Evolution of the overall error E in a two-layer feed-forward network, trained by error backpropagation (with $\epsilon K = 0.01$, $K = 10$ inputs, and a single output neuron). The results refer to independent experiments involving either a linearly separable task (with random teacher vector, lower curves) or the parity operation (upper curves), with $L \in \{2,\ 10,\ 15\}$, following random initialisation.

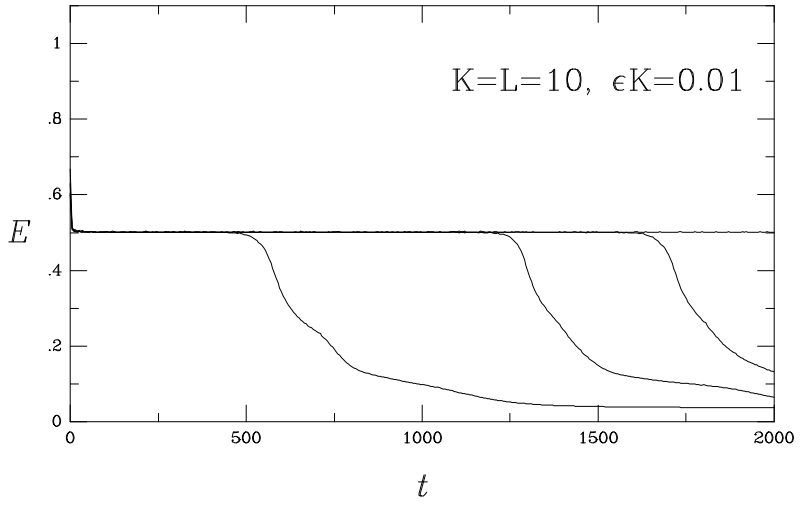


Figure 2.9: Evolution of the overall error E in a two-layer feed-forward network, trained by error backpropagation on the parity operation (with $\epsilon K = 0.01$, $K = 10$ inputs, $L = 10$ hidden neurons, and a single output neuron), following random initialisation.

Chapter 3

Recurrent Networks with Binary Neurons

We now turn to recurrent networks of binary (Ising) neurons $\sigma_i \in \{-1, 1\}$, with fixed synaptic interactions $\{J_{ij}\}$ and fixed thresholds $\{w_i\}$. Although we still have explicit (stochastic) rules (1.28) that determine the evolution in time of our system

$$\sigma_i(t+\Delta) = \ \mathrm{sgn}\,[h_i(t)+Tz_i(t)] \qquad h_i(t) = \sum_{k=1}^{N} J_{ik}\sigma_k(t)+w_i \tag{3.1}$$

(with the independent random noise variables $z_i(t)$), in contrast to the situation with layered networks, we can no longer write down some final state of our neurons in terms of given input signals, due to the feed-back present. Recurrent systems operate and are used in a manner fundamentally different from layered ones. We really have to solve the dynamics. Written in terms of probabilities, with $P(z)$ denoting the distribution of the noise variables, our dynamical rules become

$$\mathrm{Prob}\,[\sigma_i(t+\Delta)] = g\,[\sigma_i(t+\Delta)h_i(t)/T] \qquad g[x] = \int_{-\infty}^{x} dz\ P(z) \tag{3.2}$$

For recurrent systems we also have to specify in which order the neurons states are updated (for layered networks the update order did not make a difference). We restrict ourselves to two extreme cases for the update order, and find for symmetric noise distributions, i.e. where $P(z) = P(-z)$:

$$\text{parallel}: \qquad \mathrm{Prob}\,[\boldsymbol{\sigma}(t+\Delta)] = \prod_{i=1}^{N} g\,[\sigma_i(t+\Delta)h_i(t)/T]$$

$$\text{sequential}: \quad \begin{cases} \text{choose } i \text{ randomly from } \{1,\ldots,N\} \\ \mathrm{Prob}\,[\sigma_i(t+\Delta)] = g\,[\sigma_i(t+\Delta)h_i(t)/T] \end{cases}$$

so that upon making the simple choice

$$P(z) = \frac{1}{2}[1-\tanh^2(z)] \qquad \Rightarrow \qquad g[x] = \frac{1}{2}[1+\tanh(x)] = \frac{e^x}{2\cosh(x)} \tag{3.3}$$

we get, using $\cosh(-x) = \cosh(x)$:

$$\text{parallel}: \qquad \text{Prob}\,[\boldsymbol{\sigma}(t+\Delta)] = \frac{e^{\sum_i \sigma_i(t+\Delta)h_i(t)/T}}{\prod_i [2\cosh[h_i(t)/T]]} \tag{3.4}$$

$$\text{sequential}: \quad \begin{cases} \text{choose } i \text{ randomly from } \{1,\ldots,N\} \\ \text{Prob}\,[\sigma_i(t+\Delta)] = \frac{1}{2}[1+\sigma_i(t+\Delta)\tanh[h_i(t)/T]] \end{cases} \tag{3.5}$$

The particularly simple dependence in the above expressions on the state variables $\sigma_i(t+\Delta)$, as a result of the choice (3.3), will enable a detailed analysis later.

In order to obtain an idea of what to expect for such systems, however, we first turn to the deterministic (noiseless) case, $T = 0$.

3.1 Noiseless Recurrent Networks

Let us forget for the moment about the pathological case where for certain system states the post-synaptic potentials h_i can become zero. In such cases, $h_i(t) = 0$, we have to take the limit $T \to 0$ in the stochastic laws, which means that $\sigma_i(t+\Delta)$ is choosen at random from $\{-1, 1\}$ with equal probabilities. The dynamical rules now reduce to:

$$\text{parallel}: \qquad \sigma_i(t+\Delta) = \text{sgn}[\sum_j J_{ij}\sigma_j(t) + w_i] \quad (\forall i) \tag{3.6}$$

$$\text{sequential}: \quad \begin{cases} \text{choose } i \text{ randomly from } \{1,\ldots,N\} \\ \sigma_i(t+\Delta) = \text{sgn}[\sum_j J_{ij}\sigma_j(t) + w_i] \end{cases} \tag{3.7}$$

For parallel dynamics we generate a deterministic chain of successive network states

$$\boldsymbol{\sigma}(0) \to \boldsymbol{\sigma}(\Delta) \to \boldsymbol{\sigma}(2\Delta) \to \boldsymbol{\sigma}(3\Delta) \to \ldots$$

Since the number of different configurations is finite (2^N), at some stage we must obtain in this chain a configuration $\boldsymbol{\sigma}^\star$ that has already appeared earlier. Since the process is deterministic the subsequent configurations following $\boldsymbol{\sigma}^\star$ will be exactly the same ones that followed $\boldsymbol{\sigma}^\star$ after its earlier occurrence. Result: the deterministic network with parallel dynamics will always evolve into a limit cycle with period $\leq 2^N$.

For sequential dynamics with random selection of the neuron to be updated the above statement will not be true (a repeated occurrence of any state $\boldsymbol{\sigma}^\star$ does not permit more than probabilistic statements on the expected future path). However, if we were to choose the neurons to be sequentially updated in a fixed (as opposed to random) order, the dynamics becomes deterministic, so that the reasoning above applies and we know again that the system will evolve into a limit cycle with period $\leq 2^N$.

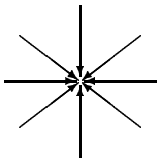
Simple Model Examples with Parallel Dynamics. We start with looking at (and solving) a number of simple recurrent model examples with parallel dynamics, of increasing complexity. The system is prepared in some initial microscopic state $\boldsymbol{\sigma}(0)$. For simplicity we choose $\Delta = 1$ (the duration of the elementary time-steps), so $t \in \{0, 1, 2, \ldots\}$.

Example (i)**:** $J_{ij} = 0$**,** $w_i \neq 0$

The dynamics (3.6) becomes:

$$\sigma_i(t{+}1) = \ \mathrm{sgn}[w_i] \qquad (\forall i)\ (\forall t \geq 0)$$

which gives the trivial solution $\sigma_i(t) = \ \mathrm{sgn}[w_i]$ $(\forall t > 0)$. In one time-step the system moves into a unique fixed-point attractor. Associated with an attractor is an attraction domain $\mathcal{D}$: the set of all initial configurations $\sigma(0)$ that are attracted by it. The size $|\mathcal{D}|$ is the number of configurations in $\mathcal{D}$. Here $\mathcal{D} = \{-1, 1\}^N$, $|\mathcal{D}| = 2^N$.



Example (ii)**:** $J_{ij} = \frac{J}{N}$**,** $w_i = 0$

We choose N to be odd (so that the average activity can never be zero). The dynamics (3.6) becomes:

$$\sigma_i(t{+}1) = \ \mathrm{sgn}[J]\ \mathrm{sgn}[\frac{1}{N}\sum_j \sigma_j(t)] \qquad (\forall i)\ (\forall t \geq 0)$$

We introduce the average activity $m(t) = \frac{1}{N}\sum_j \sigma_j(t)$, in terms of which the dynamics simplifies to

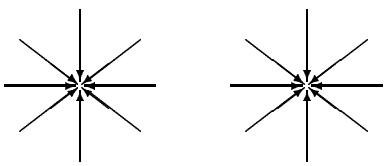
$$m(t{+}1) = \ \mathrm{sgn}[J]\ \mathrm{sgn}[m(t)] \quad (\forall i)\ (\forall t \geq 0)$$

We have to distinguish between the two cases $J > 0$ and $J < 0$:

$$J > 0: \quad m(t) = \ \mathrm{sgn}[m(0)] \qquad (\forall t > 0)$$

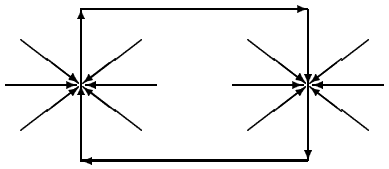
$$J < 0: \quad m(t) = (-1)^t\ \mathrm{sgn}[m(0)] \quad (\forall t > 0)$$

Note that $m(t) = 1$ implies $\boldsymbol{\sigma}(t) = (1, \ldots, 1)$, and that $m(t) = -1$ implies $\boldsymbol{\sigma}(t) = (-1, \ldots, -1)$. For $J > 0$ the system moves in one time-step into one of two fixed-point attractors: $\boldsymbol{\sigma}^+ = (1, \ldots, 1)$, with $\mathcal{D}^+ = \{\boldsymbol{\sigma} \in \{-1, 1\}^N |\ \frac{1}{N}\sum_i \sigma_i > 0\}$, and $\boldsymbol{\sigma}^- = (-1, \ldots, -1)$, with $\mathcal{D}^- = \{\boldsymbol{\sigma} \in \{-1, 1\}^N |\ \frac{1}{N}\sum_i \sigma_i < 0\}$:

$J > 0:$ 

For $J < 0$ the system moves in one time-step into a unique period-2 limit cycle attractor, $\boldsymbol{\sigma}^+ \to \boldsymbol{\sigma}^- \to \boldsymbol{\sigma}^+ \to \boldsymbol{\sigma}^- \to \ldots$, with $\mathcal{D} = \{-1, 1\}^N$:

$J < 0$:



Example (*iii*)**:** $J_{ij} = \frac{J}{N}$**,** $w_i = w \neq 0$

In order to make sure that the thresholds will not completely dominate the behaviour, we choose $|w| < |J|$. The dynamics (3.6) becomes:

$$\sigma_i(t+1) = \ \mathrm{sgn}[\frac{J}{N}\sum_j \sigma_j(t) + w] \qquad (\forall i)\ (\forall t \geq 0)$$

In terms of the average activity $m(t) = \frac{1}{N}\sum_j \sigma_j(t)$ we then find:

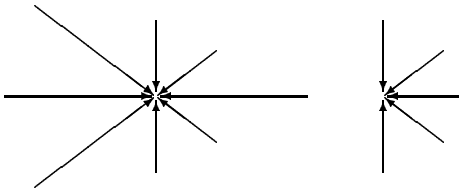
$$m(t+1) = \ \mathrm{sgn}[Jm(t) + w] \qquad (\forall t \geq 0)$$

Let us again distinguish between $J > 0$ and $J < 0$:

$$J > 0: \qquad m(t+1) = \ \mathrm{sgn}[m(t) + w/J] \qquad \begin{cases} m(0) > -w/J: & m(t) = 1 \quad (t > 0) \\ m(0) < -w/J: & m(t) = -1 \quad (t > 0) \end{cases}$$

For $J > 0$ the system again moves in one time-step into one of two fixed-point attractors: $\boldsymbol{\sigma}^+ = (1, \ldots, 1)$, with $\mathcal{D}^+ = \{\boldsymbol{\sigma} \in \{-1, 1\}^N |\ \frac{1}{N}\sum_i \sigma_i > -w/J\}$, and $\boldsymbol{\sigma}^- = (-1, \ldots, -1)$, with $\mathcal{D}^- = \{\boldsymbol{\sigma} \in \{-1, 1\}^N |\ \frac{1}{N}\sum_i \sigma_i < -w/J\}$. Note, however, that the boundaries and relative sizes of the two attraction domains are now controlled by the quantity w/J (only for $w = 0$ do we recover the previous situation where the two attractors were equally strong).

$J > 0$:

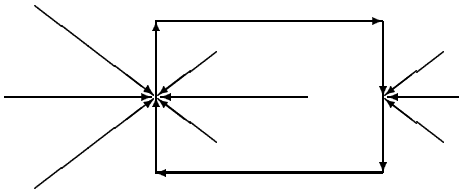


For $J < 0$, on the other hand, we find:

$$J < 0: \qquad m(t+1) = -\mathrm{sgn}[m(t) - w/|J|] \qquad \begin{cases} m(0) > -w/J: & m(t) = (-1)^t \quad (t > 0) \\ m(0) < -w/J: & m(t) = (-1)^{t+1} \quad (t > 0) \end{cases}$$

For $J < 0$ the system again moves in one time-step into the period-2 limit cycle attractor $\boldsymbol{\sigma}^+ \to \boldsymbol{\sigma}^- \to \boldsymbol{\sigma}^+ \to \boldsymbol{\sigma}^- \to \ldots$, with $\mathcal{D} = \{-1, 1\}^N$. The only difference is that we are now more likely to enter the attractor in one point than the other (controlled by w/J).

$J < 0$:



Example (iv): $J_{ij} = \frac{J}{N}$, $w_i \in \{-w, w\}$, $w \neq 0$

In order to make sure that the thresholds will not completely dominate the behaviour, we choose $|w| < |J|$. For simplicity we assume that one half of the neurons in the system have $w_i = w$, and the other half have $w_i = -w$. The dynamics (3.6) becomes:

$$\sigma_i(t+1) = \text{sgn}[\frac{J}{N}\sum_j \sigma_j(t) + w_i] \qquad (\forall i)\ (\forall t \geq 0)$$

In terms of the average activity $m(t) = \frac{1}{N}\sum_j \sigma_j(t)$ we then find:

$$m(t+1) = \frac{1}{2}\,\text{sgn}[Jm(t)+w] + \frac{1}{2}\,\text{sgn}[Jm(t)-w] \qquad (\forall t \geq 0)$$

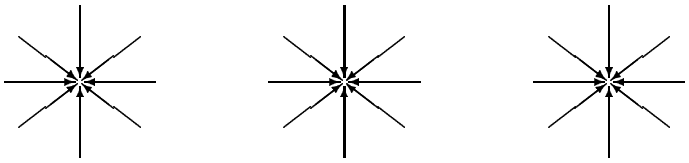
We distinguish between $J > 0$ and $J < 0$:

$$J > 0: \quad m(t+1) = \tfrac{1}{2}\,\text{sgn}[m(t)+w/J] + \tfrac{1}{2}[\,\text{sgn}[m(t)-w/J]$$

$$\begin{cases} m(0) > |w/J|: & m(t) = 1 \quad (t > 0) \\ |m(0)| < |w/J|: & m(t) = 0 \quad (t > 0) \\ m(0) < -|w/J|: & m(t) = -1 \quad (t > 0) \end{cases}$$

Note that for the stationary situation $m = 0$ we find (from the dynamic laws) $\sigma_i = \text{sgn}[w_i]$ (a microscopic fixed-point, to be denoted by $\boldsymbol{\sigma}^0$). For $J > 0$ the system moves in one time-step into one of three fixed-point attractors: $\boldsymbol{\sigma}^+ = (1, \ldots, 1)$, with $\mathcal{D}^+ = \{\boldsymbol{\sigma} \in \{-1, 1\}^N |\ \frac{1}{N}\sum_i \sigma_i > |w/J|\}$, $\boldsymbol{\sigma}^0$, with $\mathcal{D}^0 = \{\boldsymbol{\sigma} \in \{-1, 1\}^N |\ |\frac{1}{N}\sum_i \sigma_i| < |w/J|\}$, and $\boldsymbol{\sigma}^- = (-1, \ldots, -1)$, with $\mathcal{D}^- = \{\boldsymbol{\sigma} \in \{-1, 1\}^N |\ \frac{1}{N}\sum_i \sigma_i < -|w/J|\}$. The boundaries and relative sizes of the three attraction domains are now controlled by the quantity w/J; the two $m = \pm 1$ attractors are always equally strong. For $w \to 0$ the attractor $\boldsymbol{\sigma}^0$ is removed, and we return to the previously studied case with only two attractors.

$J > 0$:



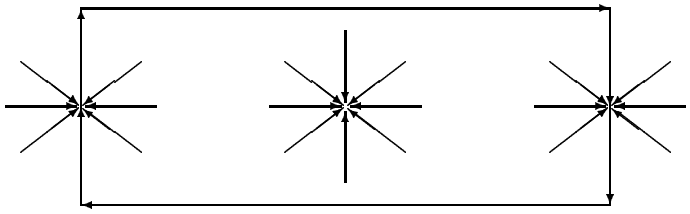
For $J<0$, on the other hand, we obtain:

$$J<0: \quad m(t+1) = -\tfrac{1}{2}\,\mathrm{sgn}[m(t)+w/J] - \tfrac{1}{2}\,\mathrm{sgn}[m(t)-w/J]$$

$$\begin{cases} m(0) > |w/J|: & m(t) = (-1)^t \quad (t>0) \\ |m(0)| < |w/J|: & m(t) = 0 \quad (t>0) \\ m(0) < -|w/J|: & m(t) = (-1)^{t+1} \quad (t>0) \end{cases}$$

For $J<0$ the system moves in one time-step either into the period-2 limit cycle attractor $\boldsymbol{\sigma}^+ \to \boldsymbol{\sigma}^- \to \boldsymbol{\sigma}^+ \to \boldsymbol{\sigma}^- \to \ldots$, with $\mathcal{D} = \{\boldsymbol{\sigma} \in -1,1\}^N |\ |\frac{1}{N}\sum_i| > |w/J|\}$, or into the fixed-point attractor $\boldsymbol{\sigma}^0$, with $\mathcal{D}^0 = \{\boldsymbol{\sigma} \in \{-1,1\}^N |\ |\frac{1}{N}\sum_i \sigma_i| < |w/J|\}$. The boundaries and relative sizes of the two attraction domains are again controlled by the quantity w/J. For $w \to 0$ the attractor $\boldsymbol{\sigma}^0$ is removed, and we return to the previously studied case with only the limit-cycle attractor.

$J<0$:



Example (v): $J_{ij} = \frac{1}{N}(\xi_i - \xi_j)$, $w_i = 0$

For simplicity we choose $\xi_i \in \{-1,1\}$ $(\forall i)$, such that half of the neurons in the system will have $\xi_i = 1$ and the other half will have $\xi_i = -1$. The dynamics (3.6) becomes:

$$\sigma_i(t+1) = \mathrm{sgn}[\xi_i \frac{1}{N}\sum_j \sigma_j(t) - \frac{1}{N}\sum_j \xi_j \sigma_j(t)] \qquad (\forall i)\ (\forall t \geq 0)$$

The relevant macroscopic quantitities to inspect now turn out to be

$$m_1(t) = \frac{1}{N}\sum_j \sigma_j(t) \qquad m_2(t) = \frac{1}{N}\sum_j \xi_j \sigma_j(t)$$

with which we can write:

$$\begin{aligned} m_1(t+1) &= \tfrac{1}{N}\textstyle\sum_i\ \mathrm{sgn}[\xi_i m_1(t) - m_2(t)] \\ &= \tfrac{1}{2}\,\mathrm{sgn}[m_1(t)-m_2(t)] - \tfrac{1}{2}\,\mathrm{sgn}[m_1(t)+m_2(t)] \\ m_2(t+1) &= \tfrac{1}{N}\textstyle\sum_i\ \mathrm{sgn}[m_1(t) - \xi_i m_2(t)] \\ &= \tfrac{1}{2}\,\mathrm{sgn}[m_1(t)-m_2(t)] + \tfrac{1}{2}\,\mathrm{sgn}[m_1(t)+m_2(t)] \end{aligned} \qquad (\forall t \geq 0)$$

At this stage it is convenient to switch to the new variables:

$$m_\pm(t) = m_1(t) \pm m_2(t), \qquad m_1(t) = \frac{1}{2}[m_+(t)+m_-(t)], \qquad m_2(t) = \frac{1}{2}[m_+(t)-m_-(t)]$$

which leads to the simple equations:

$$m_+(t+1) = \text{sgn}[m_-(t)] \qquad m_-(t+1) = -\text{sgn}[m_+(t)]$$

From this it follows, in turn, that:

$$\begin{pmatrix} m_+(t+2) \\ m_-(t+2) \end{pmatrix} = -\begin{pmatrix} \text{sgn}[m_+(t)] \\ \text{sgn}[m_-(t)] \end{pmatrix} \qquad \Rightarrow \qquad \begin{pmatrix} m_+(t+4) \\ m_-(t+4) \end{pmatrix} = \begin{pmatrix} \text{sgn}[m_+(t)] \\ \text{sgn}[m_-(t)] \end{pmatrix}$$

$$\Rightarrow \begin{pmatrix} m_+(t+8) \\ m_-(t+8) \end{pmatrix} = \begin{pmatrix} m_+(t+4) \\ m_-(t+4) \end{pmatrix}$$

In at most four time-steps the system will have entered a period-4 limit cycle solution. If $m_+(0) \neq 0$ and $m_-(0) \neq 0$, this solution is

$$\begin{pmatrix} m_+ \\ m_- \end{pmatrix}: \qquad \begin{pmatrix} 1 \\ 1 \end{pmatrix} \to \begin{pmatrix} 1 \\ -1 \end{pmatrix} \to \begin{pmatrix} -1 \\ -1 \end{pmatrix} \to \begin{pmatrix} -1 \\ 1 \end{pmatrix} \to \begin{pmatrix} 1 \\ 1 \end{pmatrix} \to \ldots$$

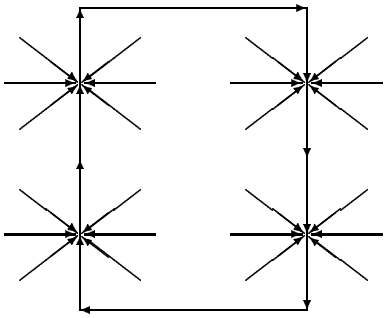
Translated back into m_1 and m_2 this implies:

$$\begin{pmatrix} m_1 \\ m_2 \end{pmatrix}: \qquad \begin{pmatrix} 1 \\ 0 \end{pmatrix} \to \begin{pmatrix} 0 \\ 1 \end{pmatrix} \to \begin{pmatrix} -1 \\ 0 \end{pmatrix} \to \begin{pmatrix} 0 \\ -1 \end{pmatrix} \to \begin{pmatrix} 1 \\ 0 \end{pmatrix} \to \ldots$$

Note that $m_1 = \pm 1$ means $\boldsymbol{\sigma} = \pm\boldsymbol{\sigma}^+ = \pm(1, \ldots, 1)$, and that $m_2 = \pm 1$ means $\boldsymbol{\sigma} = \pm\boldsymbol{\xi}$. As a result we find that at the microscopic level we have the period-4 limit-cycle attractor:

$$\boldsymbol{\sigma}^+ \to \boldsymbol{\xi} \to -\boldsymbol{\sigma}^+ \to -\boldsymbol{\xi} \to \boldsymbol{\sigma}^+ \to \ldots$$

The remaining periodic solutions, obtained when at least one of the $m_\pm(0)$ is zero, turn out to be unstable. The resulting picture is therefore



Example (vi)**:** $J_{ij} = \delta_{\pi(i),j}$**,** $w_i = 0$

Our final simple example illustrates the occurrence of even larger periods. Here π is a mapping on the set of neuron indices:

$$\pi: \quad \{1, \ldots, N\} \to \{1, \ldots, N\}$$

The dynamical rules (3.6) reduce to

$$\sigma_i(t+1) = \text{sgn}[\sigma_{\pi(i)}(t)] = \sigma_{\pi(i)}(t) \qquad (\forall i)\ (\forall t \geq 0)$$

At each time-step each neuron i determines its new state by simply copying the present state of some specific colleague $\pi(i)$. One can find quite a wide range of periods, by variation of the index operation π. Some examples:

- $\pi(i) = k$ $(\forall i)$. All neurons copy their new state from the same location k. This leads to
$$\sigma_i(t+1) = \sigma_k(t) \ (\forall i) \quad \Rightarrow \quad \sigma_i(t) = \sigma_k(0) \ (\forall i) \ (\forall t > 0)$$
The system ends up in one of two fixed-point attractors: $\boldsymbol{\sigma}^+ = (1, \ldots, 1)$, with $\mathcal{D}^+ = \{\boldsymbol{\sigma} \in \{-1, 1\} | \ \sigma_k(0) = 1\}$ and $\boldsymbol{\sigma}^- = (-1, \ldots, -1)$, with $\mathcal{D}^- = \{\boldsymbol{\sigma} \in \{-1, 1\} | \ \sigma_k(0) = -1\}$.
- $\pi(i) = N+1-i$ $(\forall i)$. Note that π now obeys $\pi(\pi(i)) = \pi(N+1-i) = i$ $(\forall i)$. As a result the dynamics is itself 2-periodic:
$$\sigma_i(t+1) = \sigma_{N+1-i}(t) \ (\forall i) \quad \Rightarrow \quad \begin{cases} \sigma_i(t) = \sigma_i(0) & (\forall i) \ (\forall t > 0 \text{ even}) \\ \sigma_i(t) = \sigma_{N+1-i}(0) & (\forall i) \ (\forall t > 0 \text{ odd}) \end{cases}$$
There are many fixed-points (corresponding to initial states with $\sigma_i(0) = \sigma_{N+1-i}(0)$ for all i), and period-2 cycles, which together cover the whole state space $\{-1, 1\}^N$.
- $\pi(i) = i+1 \ (\text{mod } N)$ $(\forall i)$. This is just an (N-periodic) index shift. One finds
$$\sigma_i(t+1) = \sigma_{i+1}(t) \ (i: \ \text{mod } N) \ (\forall i) \quad \Rightarrow \quad \sigma_i(t) = \sigma_{i+t}(0) \ (\forall i) \ (\forall t > 0)$$
Here we find many limit cycles with periods up to N (some have smaller periods due to periodicities in $\boldsymbol{\sigma}(0)$; for instance, $\boldsymbol{\sigma}(0) = (1, \ldots, 1)$ gives a period 1).

The Role of Synaptic Symmetry: Lyapunov Functions. It is clear that the diversity in the possible modes of operation in these systems is quite large. If we were to repeat the above exercise of solving simple models for the case of sequential dynamics, we would again find qualitative differences. We clearly need some tools for classification. It turns out that a rather relevant feature by which we should distinguish between various systems is whether or not the matrix of synaptic interactions is symmetric, i.e. whether $J_{ij} = J_{ji}$ for all (i, j). As always we exclude the pathological cases where local fields $h_i(\boldsymbol{\sigma})$ can be *exactly* zero.

Fact: If the synaptic matrix $\boldsymbol{J}$ is symmetric, i.e. $J_{ij} = J_{ji}$ for all (ij), then the quantity
$$L(\boldsymbol{\sigma}) = -\sum_i |\sum_j J_{ij}\sigma_j + w_i| - \sum_i \sigma_i w_i \tag{3.8}$$
is a Lyapunov function for the deterministic parallel dynamics
$$\sigma_i(t+1) = \text{sgn}[\sum_j J_{ij}\sigma_j + w_i] \quad (\forall i) \tag{3.9}$$
and the system will evolve into a period-2 limit-cycle.

Proof: Clearly $L(\boldsymbol{\sigma})$ is bounded from below: $L(\boldsymbol{\sigma}) \geq -\sum_{ij}|J_{ij}| - 2\sum_i |w_i|$. The non-trivial part of the proof is to show that it decreases monotonically with time, and that it implies evolution towards a period-2 limit cycle. Consider a transition $\boldsymbol{\sigma} \to \boldsymbol{\sigma}'$, described by the dynamical rules (3.9). The resulting change in L is given by:

$$\Delta L = L(\boldsymbol{\sigma}') - L(\boldsymbol{\sigma}) = -\sum_i |\sum_j J_{ij}\sigma'_j + w_i| + \sum_i |\sum_j J_{ij}\sigma_j + w_i| + \sum_i w_i[\sigma_i - \sigma'_i]$$

use (3.9):

$$\Delta L = -\sum_i |\sum_j J_{ij}\sigma'_j + w_i| + \sum_i \sigma'_i \left[\sum_j J_{ij}\sigma_j + w_i\right] + \sum_i w_i[\sigma_i - \sigma'_i]$$

$$= -\sum_i |\sum_j J_{ij}\sigma'_j + w_i| + \sum_{ij} \sigma'_i J_{ij}\sigma_j + \sum_i w_i\sigma_i$$

use interaction symmetry:

$$\Delta L = -\sum_i |\sum_j J_{ij}\sigma'_j + w_i| + \sum_{ij} \sigma_i J_{ij}\sigma'_j + \sum_i w_i\sigma_i$$

$$= -\sum_i |\sum_j J_{ij}\sigma'_j + w_i| + \sum_i \sigma_i \left[\sum_j J_{ij}\sigma'_j + w_i\right]$$

$$= -\sum_i |\sum_j J_{ij}\sigma'_j + w_i| \left\{1 - \sigma_i \ \mathrm{sgn}[\sum_j J_{ij}\sigma'_j + w_i]\right\} \leq 0 \qquad (3.10)$$

Note that $\Delta L \neq 0$ implies that $\Delta L \leq -\kappa$, where

$$\kappa = \min_{\boldsymbol{\sigma}} \min_i |\sum_j J_{ij}\sigma_j + w_i| > 0$$

So L decreases monotonically until at some time $t^\star < \infty$ a stage is reached where $L(\boldsymbol{\sigma}(t+1)) = L(\boldsymbol{\sigma}(t))\ \forall t > t^\star$. From then onwards it follows from (3.10) that

$$\forall i,\ \forall t > t^\star: \quad \sigma_i(t) = \mathrm{sgn}[\sum_j J_{ij}\sigma_j(t+1) + w_i] = \sigma_i(t+2)$$

which completes the proof.

For sequential dynamics we find that, apart from requiring synaptic symmetry, we need to impose an additional requirement to construct a Lyapunov function: $J_{ii} \geq 0$ $(\forall i)$. Under these conditions we can prove the following:

Fact: If the synaptic matrix $\boldsymbol{J}$ is symmetric, i.e. $J_{ij} = J_{ji}$ for all (ij), and if $J_{ii} \geq 0$ for all i, then the quantity

$$L(\boldsymbol{\sigma}) = -\frac{1}{2}\sum_{ij} \sigma_i J_{ij}\sigma_j - \sum_i \sigma_i w_i \qquad (3.11)$$

is a Lyapunov function for the sequential dynamics

$$\begin{aligned} \sigma_i(t+1) &= \ \mathrm{sgn}[\textstyle\sum_j J_{ij}\sigma_j(t)+w_i] \\ \sigma_k(t+1) &= \sigma_k(t) \quad (\forall k \neq i) \end{aligned} \tag{3.12}$$

and the system will evolve into a fixed-point. The sites $i(t)$ to be updated at time t can be drawn at random or in a fixed order (the order is not relevant for the proof).

Proof: Clearly $L(\boldsymbol{\sigma})$ is bounded from below: $L(\boldsymbol{\sigma}) \geq -\frac{1}{2}\sum_{ij}|J_{ij}|-\sum_i|w_i|$. If the neuron to be updated does not change its state, $\sigma_i' = \sigma_i$, then L will obviously remain the same. Now, on the other hand, consider a transition $\boldsymbol{\sigma} \to \boldsymbol{\sigma'}$, described by the dynamical rules (3.12), in which $\sigma_i' = -\sigma_i$:

$$\sigma_k' = \sigma_k(1-2\delta_{ik}) \qquad \sigma_i = -\,\mathrm{sgn}[\sum_j J_{ij}\sigma_j+w_i] \tag{3.13}$$

The resulting change in L is given by:

$$\Delta L = L(\boldsymbol{\sigma'}) - L(\boldsymbol{\sigma}) = -\frac{1}{2}\sum_{kl} J_{kl}\,[\sigma_k'\sigma_l'-\sigma_k\sigma_l]-\sum_k w_k[\sigma_k'-\sigma_k]$$

use (3.13):

$$\Delta L = -\frac{1}{2}\sum_{kl} J_{kl}\sigma_k\sigma_l\,[(1-2\delta_{ik})(1-2\delta_{il})-1] + 2w_i\sigma_i$$

$$= \sigma_i\sum_l J_{il}\sigma_l + \sigma_i\sum_k J_{ki}\sigma_k - 2J_{ii} + 2w_i\sigma_i$$

use interaction symmetry and sign restriction on self-interactions:

$$\Delta L = 2\sigma_i\left[\sum_j J_{ij}\sigma_j+w_i\right] - 2J_{ii} = -2|\sum_j J_{ij}\sigma_j+w_i| - 2J_{ii} < 0 \tag{3.14}$$

Note that $\Delta L \neq 0$ implies that $\Delta L \leq -\kappa$, where

$$\kappa = 2\min_{\boldsymbol{\sigma}}\min_i\left\{|\sum_j J_{ij}\sigma_j + w_i| + J_{ii}\right\} > 0$$

So L decreases monotonically until at some time $t^\star < \infty$ a stage is reached where $L(\boldsymbol{\sigma}(t+1)) = L(\boldsymbol{\sigma}(t))\ \forall t > t^\star$. From then onwards it follows from (3.14) that for every site i to be updated: $\sigma_i' = \sigma_i$, which implies

$$\forall i,\ \forall t > t^\star: \quad \sigma_i(t) = \ \mathrm{sgn}[\sum_j J_{ij}\sigma_j(t)+w_i]$$

which completes the proof.

One might think that the need for excluding negative self-interactions in the above proof is just an artifact of the particular Lyapunov used, and that one might in principle also be able to prove that sequential systems with negative self-interactions evolve to a fixed-point (by using an alternative method). This is not the case, as a simple example illustrates. Consider neurons with negative self-interactions only: $J_{ij} = J\delta_{ij}$, $J < 0$, $w_i = 0$.

$$\sigma_i(t+1) = \ \mathrm{sgn}[J]\sigma_i(t)$$
$$\sigma_k(t+1) = \sigma_k(t) \quad (\forall k \neq i)$$

Since $J < 0$, each update will result in a state change $\sigma_i' = -\sigma_i$, for all times. Therefore the absence of negative self-interactions is indeed a relevant factor in determining whether or not the sequential systems will evolve towards a fixed-point.

Information Processing in Recurrent Neural Networks. Let us now turn to the question of how recurrent networks can actually be used to process information. The basic recipe is the creation of attractors in the space $\{-1, 1\}^N$ of network states, through appropriate modification of synaptic interactions and thresholds (i.e. through 'learning'). The previous model examples gave us an impression of which types of dynamical behaviour can be obtained upon making specific choices for the system parameters. Now we are interested in the inverse problem: given a required operation, how should we choose (or modify) the parameters ?

The simplest class of attractors are fixed-points. Let us first illustrate how, through the creation of fixed-point attractors, recurrent networks can be used as so-called 'associative memories', for storing patterns (words, pictures, abstract relations, whatever). The basic ideas generalise in a natural way to the more general case of creating limit-cycle attractors of arbitrary length, in order to store pattern sequences.

- Represent each of the p items or patterns to be stored (pictures, words, etc.) as an N-bit vector $\boldsymbol{\xi}^\mu = (\xi_1^\mu, \ldots, \xi_N^\mu) \in \{-1, 1\}^N$, $\mu = 1, \ldots, p$.
- Construct synaptic interactions $\{J_{ij}\}$ and thresholds $\{w_i\}$ such that fixed-point attractors are created at the p locations of the pattern vectors $\boldsymbol{\xi}^\mu$ in state space.
- If now we are given an input to be recognised, we choose this input to be the initial microscopic network configuration $\boldsymbol{\sigma}(0)$. From this initial state the neuron state vector $\boldsymbol{\sigma}(t)$ is allowed to evolve in time autonomously, driven by the network dynamics, which will by definition lead to the nearest attractor (in some topological sense).
- The final state reached $\boldsymbol{\sigma}(\infty)$ can be interpreted as the pattern recognised by network from the input $\boldsymbol{\sigma}(0)$.

It is far from clear a priori whether this can actually be done. For such a program to work, we need to be able to create systems with many attractors with non-zero attraction domains. Furthermore, in biology (and to some degree also in engineering) we are constrained in our choice of learning rules, in the sense that only 'local' rules will be realisable and/or cost-effective. Local rules are modification recipes that involve only information available at the junction or neuron that is updated:

$$\Delta J_{ij} = \mathcal{F}[J_{ij}, \sigma_i, \sigma_j; h_i; w_i, w_j] \qquad \Delta w_i = \mathcal{G}[\sigma_i; h_i; w_i]$$

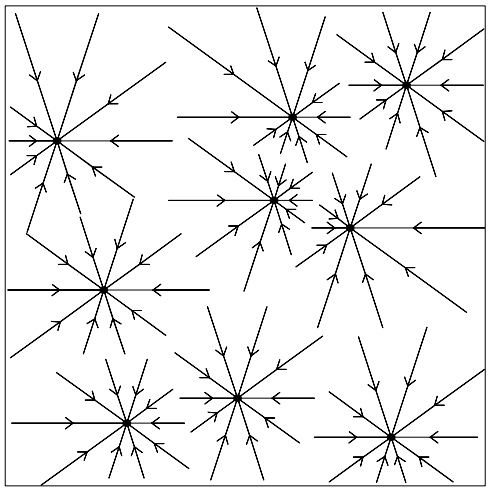


Figure 3.1: Information storage and retrieval in recurrent neural networks through the creation of attractors in phase space. Patterns $\boldsymbol{\xi}^\mu$ to be retrieved: $\bullet$. If the interactions are choosen to be symmetric, the attractors will be fixed-points or period-2 limit-cycles.

Finally, it is clear that the basic idea will in due course need some refinement. For instance, if only the p patterns to be stored are attractors, each initial state will eventually lead to pattern recognition (also nonsensical or random ones), so we will also need an attractor to act as a rubbish bin (attracting all initial states we would not like to see recognised).

To illustrate the fundamental features of the idea, let us first consider the simplest case and try to store just a single pattern $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N) \in \{-1, 1\}^N$ in a noiseless fully recurrent network with *tabula rasa* initial wiring, $J_{ij} = 0$ $(\forall ij)$, and uniform thresholds. A biologically realistic rule for interaction modification is the so-called Hebbian rule:

$$\Delta J_{ij} \sim \xi_i \xi_j \tag{3.15}$$

In words: if two neurons are required to be in the same state $(\xi_i = \xi_j)$ we increase their mutual interaction strength J_{ij}, otherwise $(\xi_i = -\xi_j)$ we decrease J_{ij}. The rule is also similar to the perceptron learning rule, the only difference being that in the case of the perceptron we only modify if the system makes an error. The resulting network is (with appropriate scaling):

$$J_{ij} = \frac{1}{N}\xi_i \xi_j \qquad w_i = w < 0, \tag{3.16}$$

We introduce new variables $\tau_i = \xi_i \sigma_i$ and $v_i = \xi_i w$, in terms of which the parallel dynamical laws can be written as

$$\tau_i(t+\Delta) = \ \mathrm{sgn}[\frac{1}{N}\sum_j \tau_j(t) + v_i] \quad (\forall i) \tag{3.17}$$

with $v_i \in \{-w, w\}$. This is precisely example (iv) studied earlier, with $J = 1$. If the fraction of neurons with $\xi_i = 1$ equals the fraction of neurons with $\xi_i = -1$, we can use the result

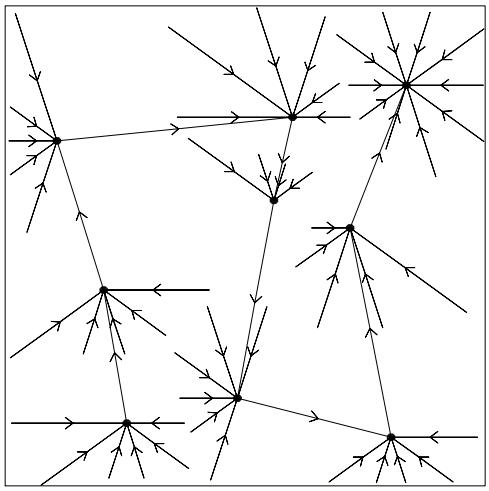


Figure 3.2: Information storage and retrieval in recurrent neural networks through the creation of attractors in phase space. Sequence of patterns $\boldsymbol{\xi}^{\mu}$ to be retrieved: $\bullet$. In order to store pattern sequences of length > 2, the interactions will have to be non-symmetric.

obtained for example (iv). Translated back into the original variables σ_i this leads to, with $m = \frac{1}{N}\sum_i \xi_i \sigma_i$:

$$\begin{aligned} m(0) > |w| : &\quad m(t) = 1 \quad (t > 0) \\ |m(0)| < |w| : &\quad m(t) = 0 \quad (t > 0) \\ m(0) < -|w| : &\quad m(t) = -1 \quad (t > 0) \end{aligned}$$

For $m = \pm 1$ we have $\sigma_i = \pm \xi_i$ $(\forall i)$; for $m = 0$ we have, according to the dynamic laws, $\sigma_i = \text{ sgn}[w] = -1$. At a microscopic level the picture thereby becomes:

$$\begin{aligned} \tfrac{1}{N}\textstyle\sum_i \xi_i \sigma_i(0) > |w| : &\quad \boldsymbol{\sigma}(t) = \boldsymbol{\xi} \quad (t > 0) \\ -|w| < \tfrac{1}{N}\textstyle\sum_i \xi_i \sigma_i(0) < |w| : &\quad \boldsymbol{\sigma}(t) = (-1, \ldots, -1) \quad (t > 0) \\ \tfrac{1}{N}\textstyle\sum_i \xi_i \sigma_i(0) < -|w| : &\quad \boldsymbol{\sigma}(t) = -\boldsymbol{\xi} \quad (t > 0) \end{aligned}$$

The system can indeed reconstruct dynamically the original pattern $\boldsymbol{\xi}$ from an input vector $\boldsymbol{\sigma}(0)$. Note that the system only reconstructs the pattern if the initial state shows a sufficient resemblance; a random initial state will lead to an attractor with zero activity. What we also note, however, is that *en passant* we have created an additional attractor: the state $-\boldsymbol{\xi}$.

The boundaries and relative sizes of the three attraction domains are controlled by the threshold w; the two $m = \pm 1$ attractors are always equally strong (this will be different if the fraction of neurons with $\xi_i = 1$ is different from the fraction with $\xi_i = -1$). For $w \to 0$ the 'rubbish bin' attractor $(-1, \ldots, -1)$ is removed. For sequential dynamics the picture is qualitatively the same, the only difference is that the various attractors are not reached in a single time-step, but are approached gradually.

Finally, let us investigate what happens if we attempt to store more than just a single pattern and apply the Hebbian learning rule (3.15) to a set of p patterns $\{\boldsymbol{\xi}^\mu\}$, with $\boldsymbol{\xi}^\mu = (\xi_1^\mu, \ldots, \xi_N^\mu)$ $(\mu = 1, \ldots, p)$. Let us, furthermore, assume for simplicity that these patterns are mutually orthogonal:

$$\frac{1}{N}\sum_i \xi_i^\mu \xi_i^\nu = \delta_{\mu\nu}$$

(which obviously requires $p \leq N$). Let us choose sequential dynamics and remove the self-interactions ($J_{ii} \to 0$), the resulting model is the so-called Hopfield model (for simplicity we take $w_i = 0\ \forall i$):

$$J_{ij} = \frac{1}{N}[1-\delta_{ij}]\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu \qquad w_i = 0 \tag{3.18}$$

We can now show that all p patterns must be stationary states of the dynamics. The Lyapunov function(3.11) can be written as:

$$L(\boldsymbol{\sigma}) = \frac{1}{2}p - \frac{1}{2}\sum_{\mu=1}^{p}\left[\frac{1}{\sqrt{N}}\sum_i \xi_i^\mu \sigma_i\right]^2$$

We can choose the normalised pattern vectors as orthogonal and normalised basis vectors in the space $\mathcal{R}^N$, $\hat{\boldsymbol{e}}^\mu = \frac{1}{\sqrt{N}}\boldsymbol{\xi}^\mu$, and use the general relation $\boldsymbol{x}^2 \geq \sum_\mu [\hat{\boldsymbol{e}}^\mu \cdot \boldsymbol{x}]^2$ to obtain a lower bound for the Lyapunov function:

$$L(\boldsymbol{\sigma}) \geq \frac{1}{2}p - \frac{1}{2}\boldsymbol{\sigma}^2 = \frac{1}{2}[p - N]$$

On the other hand, if we choose $\boldsymbol{\sigma}$ to be one of the patterns, $\boldsymbol{\sigma} = \boldsymbol{\xi}^\mu$ for some μ, we satisfy exactly the lower bound:

$$L(\boldsymbol{\xi}^\mu) = \frac{1}{2}[p - N]$$

Since any state change would decrease the value of L (which here is clearly impossible), we find that each of the patterns must correspond to a fixed-point of the dynamics. It will turn out that they are also attractors. However, we will find that additional attractors are created by the Hopfield recipe (3.18), which correspond to mixtures of the p patterns. These can be eliminated by adding noise to the dynamics.

3.2 Stochastic Recurrent Networks

Simulation Examples. We will first illustrate with numerical simulations the functioning of the Hopfield model (3.18) as an associative memory, and the description of the pattern recall process in terms of so-called overlaps:

$$m_\mu(\boldsymbol{\sigma}) = \frac{1}{N}\sum_i \xi_i^\mu \sigma_i \tag{3.19}$$

Our simulated system is an $N = 841$ Hopfield model, in which $p = 10$ patterns have been stored (see figure 3.3) according to the prescription (3.18). The two-dimensional arrangement of the neurons in this example is just a guide to the eye; since the model is fully connected, the

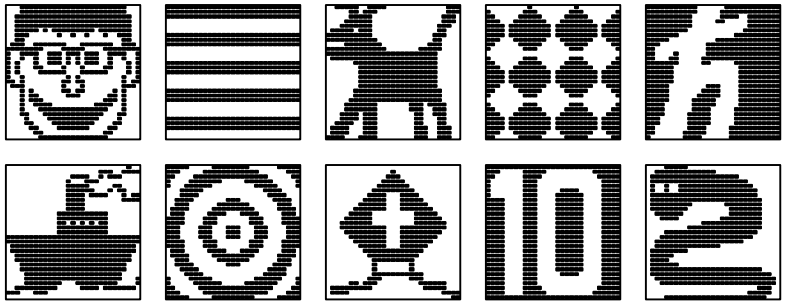


Figure 3.3: Information storage with the Hopfield model: $p = 10$ patterns represented as specific microscopic configurations in an $N = 841$ network.

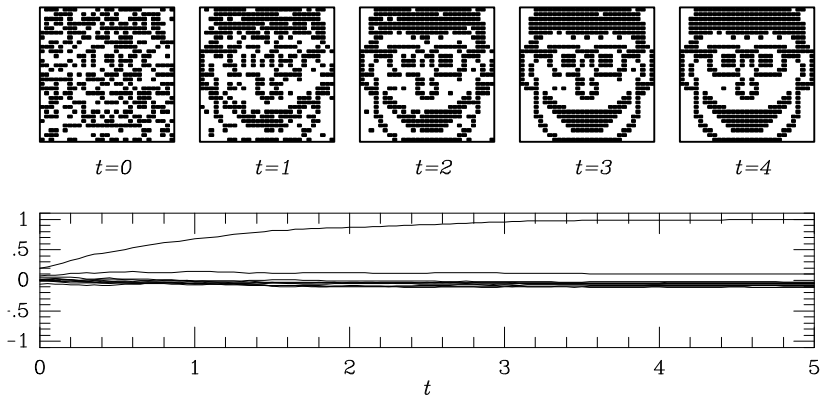


Figure 3.4: Dynamic reconstruction at $T = 0.1$ of a stored pattern from an initial state which is a corrupted version thereof. Top row: snapshots of the microscopic system state at times $t = 0, 1, 2, 3, 4$ iterations/neuron. Bottom: the corresponding values of the $p = 10$ overlap order parameters as functions of time.

spatial organisation of the neurons in the network is irrelevant. The dynamics is a sequential stochastic alignment to the post-synaptic potentials $h_i(\boldsymbol{\sigma})$, as defined by the rule (3.5), for noise level $T = 0.1$.

In figure 3.4 we show the result of letting the system evolve in time from an initial state, which is a noisy version of one of the stored patterns (here 40% of the neurons were flipped). The top row of graphs shows snapshots of the microscopic configuration as the system evolves stochastically in time. The bottom row shows the values of the $p = 10$ overlaps m_μ (defined in (3.19)), measured as functions of time; the one which evolves towards 1 correponding to the pattern being reconstructed.

Figure 3.5 shows a similar experiment, here the initial state is simply drawn at random. The system subsequently evolves towards some mixture of the stored patterns. Note that the

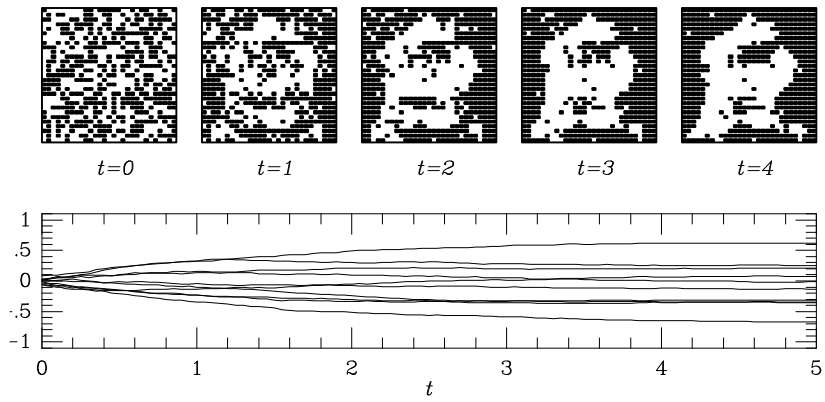


Figure 3.5: Evolution towards a spurious (mixture) state at $T = 0.1$ from a randomly drawn initial state. Top row: snapshots of the microscopic system state at times $t = 0, 1, 2, 3, 4$ iterations/neuron. Bottom: the corresponding values of the $p = 10$ overlap order parameters as functions of time.

patterns involved are significantly correlated (see figure 3.3).

Description as a Stochastic Process. Both microscopic equations (3.4,3.5) can be transformed directly into equations describing the evolution of the microscopic state probability $p_t(\boldsymbol{\sigma})$. Both take the form of a so-called Markov chain:

$$p_{t+1}(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}'} W\left[\boldsymbol{\sigma};\boldsymbol{\sigma}'\right] p_t(\boldsymbol{\sigma}') \tag{3.20}$$

For the parallel dynamics equation (3.4) we obtain the form (3.20) directly by averaging over the possible states at time t. The transition matrix is now found to be

$$W\left[\boldsymbol{\sigma};\boldsymbol{\sigma}'\right] = \prod_{i=1}^{N} \frac{e^{\beta\sigma_i h_i(\boldsymbol{\sigma}')}}{2\cosh[\beta h_i(\boldsymbol{\sigma}')]} \tag{3.21}$$

with $\beta = T^{-1}$ (the inverse noise level, $\beta = \infty$ corresponds to zero noise).

For sequential dynamics there are two types of randomness. We first have to take into account the randomness in the site to be updated. If each site is equally likely to be selected we obtain

$$p_{t+1}(\boldsymbol{\sigma}) = \frac{1}{N}\sum_{i=1}^{N}\left\{\left[\prod_{j\neq i}\delta_{\sigma_j,\sigma_j(t)}\right]\frac{1}{2}\left[1+\sigma_i\tanh[\beta h_i(\boldsymbol{\sigma}(t))]\right]\right\}$$

Note that the term $\prod_{j\neq i}\delta_{\sigma_j,\sigma_j(t)}$ precisely dictates that at time t only component σ_i is allowed to be updated. If, instead of $\boldsymbol{\sigma}(t)$, the probability distribution $p_t(\boldsymbol{\sigma})$ is given, this expression is to be averaged over the possible states at time t, with the result:

$$p_{t+1}(\boldsymbol{\sigma}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left[1+\sigma_i\tanh[\beta h_i(\boldsymbol{\sigma})]\right]p_t(\boldsymbol{\sigma}) + \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}\left[1+\sigma_i\tanh[\beta h_i(F_i\boldsymbol{\sigma})]\right]p_t(F_i\boldsymbol{\sigma}) \tag{3.22}$$

in which F_i is the i-th neuron-flip operator, defined as

$$F_i\Phi(\boldsymbol{\sigma}) \equiv \Phi(\sigma_1,\ldots,\sigma_{i-1},-\sigma_i,\sigma_{i+1},\ldots,\sigma_N)$$

The result (3.22) can be written in the form (3.20), with the transition matrix

$$W\left[\boldsymbol{\sigma};\boldsymbol{\sigma}'\right] = \delta_{\boldsymbol{\sigma},\boldsymbol{\sigma}'} + \frac{1}{N}\sum_{i=1}^{N}\left\{w_i(F_i\boldsymbol{\sigma})\delta_{\boldsymbol{\sigma},F_i\boldsymbol{\sigma}'} - w_i(\boldsymbol{\sigma})\delta_{\boldsymbol{\sigma},\boldsymbol{\sigma}'}\right\} \tag{3.23}$$

$$w_i(\boldsymbol{\sigma}) = \frac{1}{2}\left[1-\sigma_i\tanh\left[\beta h_i(\boldsymbol{\sigma})\right]\right] \tag{3.24}$$

(with $\delta_{\boldsymbol{\sigma},\boldsymbol{\sigma}'} = \prod_i \delta_{\sigma_i,\sigma_i'}$), since insertion of (3.23) into (3.20) indeed gives (3.22).

From Discrete to Continuous Times. The formal method to go from any discrete-time Markov process of the form

$$\hat{p}_{n+1}(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}'} W\left[\boldsymbol{\sigma};\boldsymbol{\sigma}'\right]\hat{p}_n(\boldsymbol{\sigma}')$$

to a continuous-time equation, is to assume in addition that the *duration* of each of the above discrete (sequential) iteration steps is a continuous random number. The statistics of these random durations are contained in the function $\pi_m(t)$, which is defined as the probability that at time t precisely m iteration steps have been made. Our new Markov process will now be described by

$$p_t(\boldsymbol{\sigma}) = \sum_{m\geq 0}\pi_m(t)\hat{p}_m(\boldsymbol{\sigma}) = \sum_{m\geq 0}\pi_m(t)\sum_{\boldsymbol{\sigma}'}W^m\left[\boldsymbol{\sigma};\boldsymbol{\sigma}'\right]p_0(\boldsymbol{\sigma}')$$

and time has become a continuous variable. For $\pi_m(t)$ we make the choice

$$\pi_m(t) \equiv \frac{1}{m!}\left(\frac{t}{\tau}\right)^m e^{-t/\tau} \tag{3.25}$$

(a Poisson distribution), with the properties

$$\frac{d}{dt}\pi_{m>0}(t) = \frac{1}{\tau}\left[\pi_{m-1}(t)-\pi_m(t)\right] \qquad \frac{d}{dt}\pi_0(t) = -\frac{1}{\tau}\pi_0(t)$$

From $\langle m\rangle_\pi = t/\tau$ it follows that τ is the average duration of a single discrete iteration step. The above choice for $\pi_m(t)$ allows us to write for the temporal derivative of $p_t(\boldsymbol{\sigma})$:

$$\tau\frac{d}{dt}p_t(\boldsymbol{\sigma}) = \sum_{m>0}\pi_{m-1}(t)\sum_{\boldsymbol{\sigma}'}W^m\left[\boldsymbol{\sigma};\boldsymbol{\sigma}'\right]p_0(\boldsymbol{\sigma}') - \sum_{m\geq 0}\pi_m(t)\sum_{\boldsymbol{\sigma}'}W^m\left[\boldsymbol{\sigma};\boldsymbol{\sigma}'\right]p_0(\boldsymbol{\sigma}')$$

$$= -p_t(\boldsymbol{\sigma}) + \sum_{\boldsymbol{\sigma}'}W\left[\boldsymbol{\sigma};\boldsymbol{\sigma}'\right]p_t(\boldsymbol{\sigma}')$$

which has the form of a so-called master equation.

This procedure can immediately be applied to the Markov chains (3.21) and (3.23). If for the sequential case in particular we also choose $\tau = \frac{1}{N}$ then in one unit of time each

neuron will on average have been updated once, and the master equation corresponding to (3.20,3.23) acquires the form

$$\frac{d}{dt}p_t(\boldsymbol{\sigma}) = \sum_{i=1}^{N}\{w_i(F_i\boldsymbol{\sigma})p_t(F_i\boldsymbol{\sigma}) - w_i(\boldsymbol{\sigma})p_t(\boldsymbol{\sigma})\} \tag{3.26}$$

In this equation the quantities $w_i(\boldsymbol{\sigma})$, defined in (3.24), have come to play the role of *transition rates*. If, instead of the rule (3.25) we just choose each sequential iteration step to have duration $\tau = 1/N$, it can be shown that equation (3.26) will describe the system state for $t \gg \frac{1}{N}$ and $N \to \infty$.

3.3 Macroscopic Analysis of Sequential Attractor Networks

In this section we show how for sequential dynamics one can calculate from the *microscopic* stochastic evolution equations (at the level of individual neurons) differential equations for the probability distribution of suitably defined *macroscopic* state variables. For mathematical convenience our starting point will be the continuous-time master equation (3.26), rather than the discrete version (3.20,3.23). We will obtain conditions for the evolution of these macroscopic state variables to (a) become deterministic in the limit of infinitely large networks and, in addition, (b) be governed by a closed set of dynamic equations. We then turn to certain classes of models and show how the macroscopic equations can be used to understand the dynamics of attractor neural networks away from saturation.

A Toy Model. Let us first illustrate the basic ideas with the help of a simple toy model:

$$J_{ij} = \frac{J}{N}\eta_i\xi_j \qquad\qquad w_i \equiv 0 \tag{3.27}$$

(the variables η_i and ξ_i are arbitrary, but may not depend on N). The interaction matrix is non-symmetric as soon as a pair (ij) exists, such that $\eta_i\xi_j \neq \eta_j\xi_i$. The local fields become $h_i(\boldsymbol{\sigma}) = J\eta_i m(\boldsymbol{\sigma})$ with $m(\boldsymbol{\sigma}) \equiv \frac{1}{N}\sum_k \xi_k\sigma_k$. Since they depend on the microscopic state $\boldsymbol{\sigma}$ only through the value of m, the latter quantity appears to constitute a natural macroscopic level of description. The probability of finding the macroscopic state $m(\boldsymbol{\sigma}) = m$ is given by

$$\mathcal{P}_t[m] = \sum_{\boldsymbol{\sigma}} p_t(\boldsymbol{\sigma})\delta\left[m - m(\boldsymbol{\sigma})\right]$$

Its time derivative is obtained by inserting (3.26):

$$\frac{d}{dt}\mathcal{P}_t[m] = \sum_{\boldsymbol{\sigma}}\sum_{k=1}^{N} p_t(\boldsymbol{\sigma})w_k(\boldsymbol{\sigma})\left\{\delta\left[m - m(\boldsymbol{\sigma}) + \frac{2}{N}\xi_k\sigma_k\right] - \delta\left[m - m(\boldsymbol{\sigma})\right]\right\}$$

$$= \frac{d}{dm}\left\{\sum_{\boldsymbol{\sigma}} p_t(\boldsymbol{\sigma})\delta\left[m - m(\boldsymbol{\sigma})\right]\frac{2}{N}\sum_{k=1}^{N}\xi_k\sigma_k w_k(\boldsymbol{\sigma})\right\} + \mathcal{O}(N^{-1})$$

Inserting the expression (3.24) for the transition rates and the local fields gives:

$$\frac{d}{dt}\mathcal{P}_t[m] = \frac{d}{dm}\left\{\mathcal{P}_t[m]\left[m - \frac{1}{N}\sum_{k=1}^{N}\xi_k\tanh[\eta_k\beta Jm]\right]\right\} + \mathcal{O}(N^{-1})$$

In the thermodynamic limit $N \to \infty$ only the first term survives. The solution of the resulting differential equation for $\mathcal{P}_t[m]$ is:

$$\mathcal{P}_t[m] = \int dm_0\ \mathcal{P}_0[m_0]\delta\left[m - m^*(t)\right]$$

$$\frac{d}{dt}m^* = \lim_{N\to\infty}\frac{1}{N}\sum_{k=1}^{N}\xi_k \tanh[\eta_k \beta J m^*] - m^* \qquad m^*(0) = m_0 \tag{3.28}$$

which can simply be verified by insertion into the differential equation for $\mathcal{P}_t[m]$. This solution describes deterministic evolution, the only uncertainty in the value of m is due to uncertainty in initial conditions. If at $t = 0$ the quantity m is known exactly, this will remain the case for finite timescales; m turns out to evolve in time according to (3.28).

Arbitrary Synaptic Interactions. Let us now allow for less trivial choices of the interaction matrix and try to calculate the evolution in time of a given set of macroscopic state variables $\mathbf{\Omega}(\boldsymbol{\sigma}) \equiv (\Omega_1(\boldsymbol{\sigma}), \ldots, \Omega_n(\boldsymbol{\sigma}))$ in the limit $N \to \infty$. The probability of finding the system in macroscopic state $\mathbf{\Omega}$ is given by:

$$\mathcal{P}_t\left[\mathbf{\Omega}\right] = \sum_{\boldsymbol{\sigma}} p_t(\boldsymbol{\sigma})\delta\left[\mathbf{\Omega} - \mathbf{\Omega}(\boldsymbol{\sigma})\right]$$

The time derivative of this distribution is obtained by inserting (3.26). If in those parts of the resulting expression which contain the operators F_i we subsequently perform transformations $\boldsymbol{\sigma} \to F_i\boldsymbol{\sigma}$, we arrive at

$$\frac{d}{dt}\mathcal{P}_t\left[\mathbf{\Omega}\right] = \sum_i \sum_{\boldsymbol{\sigma}} p_t(\boldsymbol{\sigma}) w_i(\boldsymbol{\sigma})\left\{\delta\left[\mathbf{\Omega} - \mathbf{\Omega}(F_i\boldsymbol{\sigma})\right] - \delta\left[\mathbf{\Omega} - \mathbf{\Omega}(\boldsymbol{\sigma})\right]\right\}$$

Upon writing $\Omega_\mu(F_i\boldsymbol{\sigma}) = \Omega_\mu(\boldsymbol{\sigma}) + \Delta_{i\mu}(\boldsymbol{\sigma})$ and making a Taylor expansion in powers of $\{\Delta_{i\mu}(\boldsymbol{\sigma})\}$, we finally obtain the so-called Kramers-Moyal expansion:

$$\frac{d}{dt}\mathcal{P}_t\left[\mathbf{\Omega}\right] = \sum_{l\geq 1}\frac{(-1)^l}{l!}\sum_{\mu_1=1}^{n}\cdots\sum_{\mu_l=1}^{n}\frac{\partial^l}{\partial\Omega_{\mu_1}\cdots\partial\Omega_{\mu_l}}\left\{\mathcal{P}_t\left[\mathbf{\Omega}\right]F^{(l)}_{\mu_1\cdots\mu_l}\left[\mathbf{\Omega};t\right]\right\} \tag{3.29}$$

which is defined in terms of conditional averages $\langle f(\boldsymbol{\sigma})\rangle_{\mathbf{\Omega};t}$ and the 'discrete derivatives' $\Delta_{j\mu}(\boldsymbol{\sigma}) = \Omega_\mu(F_j\boldsymbol{\sigma}) - \Omega_\mu(\boldsymbol{\sigma})$:

$$F^{(l)}_{\mu_1\cdots\mu_l}\left[\mathbf{\Omega};t\right] = \langle\sum_{j=1}^{N} w_j(\boldsymbol{\sigma})\Delta_{j\mu_1}(\boldsymbol{\sigma})\cdots\Delta_{j\mu_l}(\boldsymbol{\sigma})\rangle_{\mathbf{\Omega};t}$$

$$\langle f(\boldsymbol{\sigma})\rangle_{\mathbf{\Omega};t} = \frac{\sum_{\boldsymbol{\sigma}} p_t(\boldsymbol{\sigma})\delta\left[\mathbf{\Omega} - \mathbf{\Omega}(\boldsymbol{\sigma})\right] f(\boldsymbol{\sigma})}{\sum_{\boldsymbol{\sigma}} p_t(\boldsymbol{\sigma})\delta\left[\mathbf{\Omega} - \mathbf{\Omega}(\boldsymbol{\sigma})\right]}$$

The expansion (3.29) is to be interpreted in a distributional sense, i.e. only to be used in expressions of the form $\int d\mathbf{\Omega}\mathcal{P}_t(\mathbf{\Omega})G(\mathbf{\Omega})$ with sufficiently smooth functions $G(\mathbf{\Omega})$, so that all derivatives are well-defined and finite. Furthermore, (3.29) will only make sense if the discrete derivatives $\Delta_{j\mu}$, which measure the sensitivity of the macroscopic quantities to single neuron flips, are sufficiently small. This is to be expected: for finite N *any* state variable $\Omega_\mu(\boldsymbol{\sigma})$

can only assume a finite number of possible values; only in the limit $N \to \infty$ may we expect smooth probability distributions for our macroscopic quantities (the probability distribution of state variables which only depend on a *small* number of neurons, however, will *not* become smooth, whatever the system size). Retaining only the first ($l = 1$) term in the series (3.29) leads us to a Liouville equation which describes deterministic flow in $\mathbf{\Omega}$ space, driven by the flow field $\boldsymbol{F}^{(1)}$. A sufficient condition for the set $\mathbf{\Omega}(\boldsymbol{\sigma})$ to evolve in time deterministically in the limit $N \to \infty$ is therefore:

$$\lim_{N\to\infty} \sum_{l\geq 2} \frac{1}{l!} \sum_{\mu_1=1}^{n} \cdots \sum_{\mu_l=1}^{n} \sum_{j=1}^{N} \langle |\Delta_{j\mu_1}(\boldsymbol{\sigma}) \cdots \Delta_{j\mu_l}(\boldsymbol{\sigma})| \rangle_{\mathbf{\Omega};t} = 0 \tag{3.30}$$

Note: since the partial derivatives $\partial/\partial\Omega_\mu$ in the Kramers-Moyal expansion are defined in a distributional sense, i.e. in any calculation of expectation values $\langle f[\mathbf{\Omega}] \rangle$ they will be moved to the function $f[\mathbf{\Omega}]$ via integration by parts, they need not occur in the condition (3.30). In the simple case where all 'derivatives' $\Delta_{j\mu}$ are of the same order in the system size N (i.e. there is a monotonic function $\tilde{\Delta}_N$ such that $\Delta_{j\mu} = \mathcal{O}(\tilde{\Delta}_N)$ for all $j\mu$), the above criterion becomes:

$$\lim_{N\to\infty} n\tilde{\Delta}_N \sqrt{N} = 0 \tag{3.31}$$

If the condition (3.30) is satisfied we can for large N describe the evolution of the macroscopic probability density by the Liouville equation:

$$\frac{d}{dt}\mathcal{P}_t[\mathbf{\Omega}] = -\sum_{\mu=1}^{n} \frac{\partial}{\partial\Omega_\mu} \left\{ \mathcal{P}_t[\mathbf{\Omega}]\, F_\mu^{(1)}[\mathbf{\Omega};t] \right\}$$

the solution of which describes deterministic flow:

$$\mathcal{P}_t[\mathbf{\Omega}] = \int d\mathbf{\Omega}_0 \mathcal{P}_0[\mathbf{\Omega}_0]\, \delta[\mathbf{\Omega} - \mathbf{\Omega}^*(t)]$$

$$\frac{d}{dt}\mathbf{\Omega}^*(t) = \boldsymbol{F}^{(1)}[\mathbf{\Omega}^*(t);t] \qquad \mathbf{\Omega}^*(0) = \mathbf{\Omega}_0 \tag{3.32}$$

In taking the limit $N \to \infty$, however, we have to keep in mind that the result is obtained by taking this limit for *finite* t. According to (3.29) the $l > 1$ terms do come into play for sufficiently large times t; for $N \to \infty$, but these times diverge by virtue of (3.30).

Equation (3.32) will in general not be autonomous; tracing back the origin of the explicit time dependence in the right-hand side of (3.32) one finds that in order to calculate $\boldsymbol{F}^{(1)}$ one needs to know the microscopic probability distribution $p_t(\boldsymbol{\sigma})$. This, in turn, requires solving the master equation (3.26) (which is exactly what one tries to avoid). The way out is to choose the macroscopic state variables $\mathbf{\Omega}$ in such a way that there is no explicit time dependence in the flow field $\boldsymbol{F}^{(1)}[\mathbf{\Omega};t]$ (if possible). According to the definition of the flow field this implies making sure that there exists a vector field $\mathbf{\Phi}[\mathbf{\Omega}]$ such that

$$\lim_{N\to\infty} \sum_{j=1}^{N} w_j(\boldsymbol{\sigma}) \mathbf{\Delta}_j(\boldsymbol{\sigma}) = \mathbf{\Phi}[\mathbf{\Omega}(\boldsymbol{\sigma})] \tag{3.33}$$

(with $\mathbf{\Delta}_j \equiv (\Delta_{j1}, \ldots, \Delta_{jn})$) in which case the time dependence of $\boldsymbol{F}^{(1)}$ drops out and the macroscopic state vector evolves in time according to:

$$\frac{d}{dt}\mathbf{\Omega} = \mathbf{\Phi}[\mathbf{\Omega}]$$

For this method to apply, a suitable separable structure of the interaction matrix is required. If, for instance, the macroscopic state variables Ω_μ depend linearly on the microscopic state variables $\boldsymbol{\sigma}$ (i.e. $\boldsymbol{\Omega}(\boldsymbol{\sigma}) = \frac{1}{N}\sum_{j=1}^{N}\boldsymbol{\omega}_j\sigma_j$), we obtain (with the transition rates (3.24)):

$$\lim_{N\to\infty}\sum_{j=1}^{N} w_j(\boldsymbol{\sigma})\boldsymbol{\Delta}_j(\boldsymbol{\sigma}) = \lim_{N\to\infty}\frac{1}{N}\sum_{j=1}^{N}\boldsymbol{\omega}_j\tanh(\beta h_j(\boldsymbol{\sigma})) - \boldsymbol{\Omega}$$

in which case it turns out that the only further condition necessary for (3.33) to hold is that all local fields h_k must (in leading order in N) depend on the microscopic state $\boldsymbol{\sigma}$ only through the values of the *macroscopic* state variables $\boldsymbol{\Omega}$ (since the local fields depend linearly on $\boldsymbol{\sigma}$ this, in turn, implies that the interaction matrix must be separable).

Next we will show how the above formalism can be applied to networks for which the matrix of interactions J_{ij} has a separable form (which includes most symmetric and non-symmetric Hebbian type attractor models). We will restrict ourselves to models with $w_i = 0$; the introduction of non-zero thresholds is straightforward and does not pose new problems.

Separable Models: Description at the Level of Overlaps. At the macroscopic level of description of the pattern overlaps $m_\mu(\boldsymbol{\sigma}) = \frac{1}{N}\sum_i \xi_i^\mu\sigma_i$ one will find an *autonomous* set of dynamical laws if the interaction matrix is bilinear, i.e.

$$J_{ij} \equiv \frac{1}{N}\sum_{\mu\nu=1}^{p}\xi_i^\mu A_{\mu\nu}\xi_j^\nu \qquad\qquad \boldsymbol{\xi}_i \equiv (\xi_i^1,\ldots,\xi_i^p) \tag{3.34}$$

The Hopfield model corresponds to choosing $A_{\mu\nu} \equiv \delta_{\mu\nu}$ and $\xi_i^\mu \in \{-1,1\}$. The local fields h_k can now be written in terms of the overlap order parameters m_μ:

$$m_\mu(\boldsymbol{\sigma}) \equiv \frac{1}{N}\sum_{i=1}^{N}\xi_i^\mu\sigma_i \qquad h_k(\boldsymbol{\sigma}) = \boldsymbol{\xi}_k\cdot A\boldsymbol{m}(\boldsymbol{\sigma}) \qquad \boldsymbol{m} \equiv (m_1,\ldots,m_p) \tag{3.35}$$

Since for the present choice of macroscopic variables we find $\Delta_{j\mu} = \mathcal{O}(N^{-1})$, the evolution in time of the overlap vector $\boldsymbol{m}$ becomes deterministic in the limit $N\to\infty$ if (according to (3.31)):

$$\lim_{N\to\infty}\frac{p}{\sqrt{N}} = 0$$

Condition (3.33) holds, since

$$\sum_{j=1}^{N} w_j(\boldsymbol{\sigma})\Delta_{j\mu}(\boldsymbol{\sigma}) = \frac{1}{N}\sum_{k=1}^{N}\boldsymbol{\xi}_k\tanh\left[\beta\boldsymbol{\xi}_k\cdot A\boldsymbol{m}\right] - \boldsymbol{m}$$

In the limit $N\to\infty$ the evolution in time of the overlap vector $\boldsymbol{m}$ is governed by an autonomous set of differential equations; if the vectors $\boldsymbol{\xi}_k$ are drawn at random according to some distribution $\rho(\boldsymbol{\xi})$ these dynamical laws become:

$$\frac{d}{dt}\boldsymbol{m} = \langle\boldsymbol{\xi}\tanh\left[\beta\boldsymbol{\xi}\cdot A\boldsymbol{m}\right]\rangle_{\boldsymbol{\xi}} - \boldsymbol{m} \qquad \langle\Phi(\boldsymbol{\xi})\rangle_{\boldsymbol{\xi}} \equiv \int d\boldsymbol{\xi}\;\rho(\boldsymbol{\xi})\Phi(\boldsymbol{\xi}) \tag{3.36}$$

Symmetry of the interaction matrix is not required.

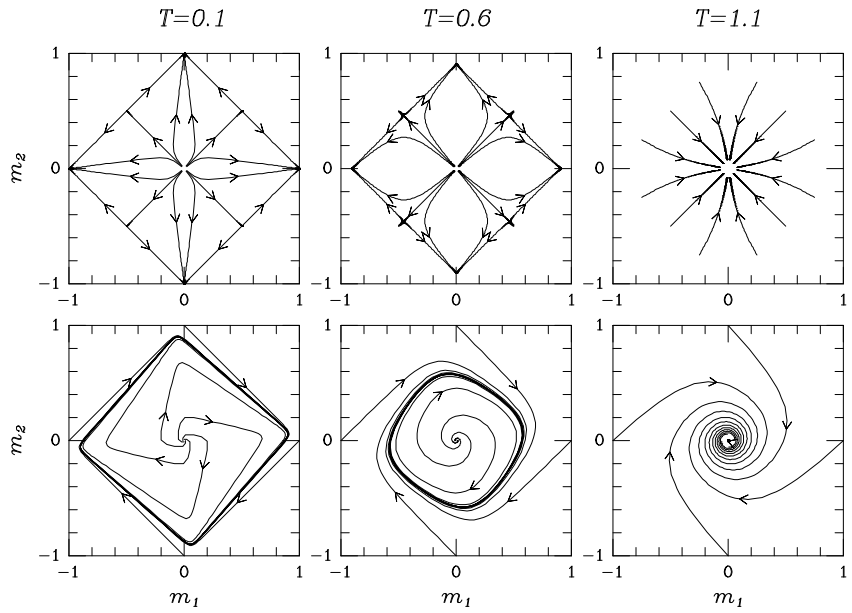


Figure 3.6: Flow diagrams obtained by numerically solving the deterministic overlap equations for $p = 2$. Upper row: $A_{\mu\nu} = \delta_{\mu\nu}$ (the Hopfield model); lower row: $A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$ (for both models the critical noise level is $T_c = 1$).

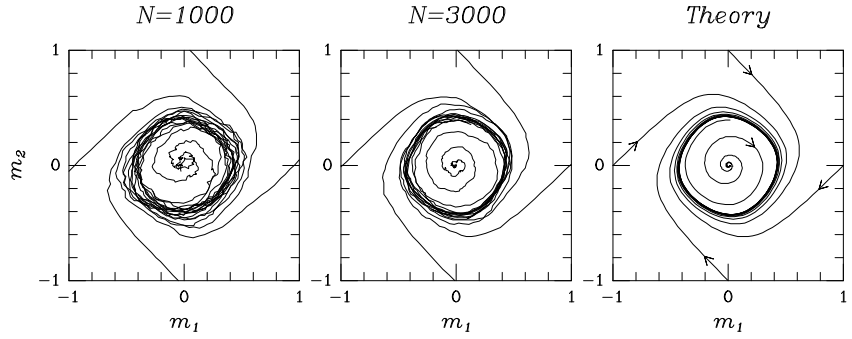


Figure 3.7: Comparison between simulation results for finite systems ($N = 1000$ and $N = 3000$) and the analytical prediction (flow equations) with respect to the evolution of the overlaps (m_1, m_2); $p = 2$, $T = 0.8$ and $A = \begin{pmatrix} 1 & 1 \\ -1 & 1 \end{pmatrix}$.

Figure 3.6 shows in the m_1, m_2-plane the result of solving the macroscopic laws (3.36) numerically for $p = 2$, randomly drawn pattern bits $\xi_i^\mu \in \{-1, 1\}$, and two choices of the matrix A. The first choice (upper row) corresponds to the Hopfield model; as the amount of noise increases, the amplitudes of the four attractors (corresponding to the two patterns $\boldsymbol{\xi}^\mu$ and their mirror images $-\boldsymbol{\xi}^\mu$) continuously decrease, until at the critical level $T_c = 1$ they merge into the trivial attractor $\boldsymbol{m} = (0, 0)$. The second choice corresponds to a non-symmetric model (i.e. without detailed balance); at the macroscopic level of desciption (at finite timescales) the system clearly does not approach equilibrium; macroscopic order now manifests itself in the form of a limit-cycle (provided the amount of noise T is below the critical level $T_c = 1$ where this limit-cycle is destroyed). To what extent the laws (3.36) are in agreement with the result of performing the actual simulations in finite systems is illustrated in figure 3.7.

3.4 Stationary States of the Hopfield Model

For the simplest separable model, the sequential Hopfield model (3.18) with finite p, we obtained in the limit $N \to \infty$:

$$\frac{d}{dt}\boldsymbol{m} = \langle \boldsymbol{\xi} \tanh\left[\beta \boldsymbol{\xi} \cdot \boldsymbol{m}\right] \rangle_{\boldsymbol{\xi}} - \boldsymbol{m} \tag{3.37}$$

We also know that the sequential dynamics laws drive the system to an equilibrium state. The stationary values of the overlaps are the solutions of

$$\boldsymbol{m} = \langle \boldsymbol{\xi} \tanh\left[\beta \boldsymbol{\xi} \cdot \boldsymbol{m}\right] \rangle_{\boldsymbol{\xi}} \tag{3.38}$$

Analysis of Stationary Overlap Equations: Mixture States. We will restrict our further discussion to the case of randomly drawn patterns, so

$$\langle \Phi(\boldsymbol{\xi}) \rangle_{\boldsymbol{\xi}} = 2^{-p} \sum_{\boldsymbol{\xi} \in \{-1,1\}^p} \Phi(\boldsymbol{\xi}), \qquad \langle \xi_\mu \xi_\nu \rangle_{\boldsymbol{\xi}} = \delta_{\mu\nu}$$

(generalisation to correlated patterns is in principle straightforward). We first establish an upper bound for the noise level $T = 1/\beta$ for non-trivial solutions $\boldsymbol{m}$ to exist, by writing (3.38) in integral form:

$$m_\mu = \beta \langle \xi_\mu \left(\boldsymbol{\xi} \cdot \boldsymbol{m}\right) \int_0^1 d\lambda \left[1 - \tanh^2\left(\beta\lambda \boldsymbol{\xi} \cdot \boldsymbol{m}\right)\right] \rangle_{\boldsymbol{\xi}}$$

from which we deduce

$$\begin{aligned} 0 &= \boldsymbol{m}^2 - \beta \langle (\boldsymbol{\xi} \cdot \boldsymbol{m})^2 \textstyle\int_0^1 d\lambda \left[1 - \tanh^2\left(\beta\lambda \boldsymbol{\xi} \cdot \boldsymbol{m}\right)\right] \rangle_{\boldsymbol{\xi}} \\ &\geq \boldsymbol{m}^2 - \beta \langle (\boldsymbol{\xi} \cdot \boldsymbol{m})^2 \rangle_{\boldsymbol{\xi}} = \boldsymbol{m}^2 \left[1 - \beta\right] \end{aligned}$$

For $T > 1$ the only solution of (3.38) is the disordered state $\boldsymbol{m} = 0$. At $T = 1$, however, a continuous bifurcation occurs, which follows from expanding (3.38) for small $|\boldsymbol{m}|$ in powers

of $\tau = \beta - 1$:

$$\begin{aligned} m_\mu &= (1+\tau)m_\mu - \tfrac{1}{3}\textstyle\sum_{\nu\rho\lambda} m_\nu m_\rho m_\lambda \langle \xi_\mu\xi_\nu\xi_\rho\xi_\lambda\rangle_{\boldsymbol{\xi}} + \mathcal{O}(\boldsymbol{m}^5, \tau\boldsymbol{m}^3) \\ &= m_\mu\left[1 + \tau - \boldsymbol{m}^2 + \tfrac{2}{3}m_\mu^2\right] + \mathcal{O}(\boldsymbol{m}^5, \tau\boldsymbol{m}^3) \end{aligned}$$

The bifurcating non-zero solutions of (3.38) scale as $m_\mu = \tilde{m}_\mu \tau^{1/2} + \mathcal{O}(\tau^{3/2})$, with for each μ:

$$\tilde{m}_\mu = 0 \quad \text{or} \quad 0 = 1 - \tilde{\boldsymbol{m}}^2 + \frac{2}{3}\tilde{m}_\mu^2$$

The solutions are of the form $\tilde{m}_\mu \in \{-\tilde{m}, 0, \tilde{m}\}$. If we denote with n the number of non-zero components in the vector $\tilde{\boldsymbol{m}}$, we derive from the above identities:

$$\tilde{m}_\mu = 0 \quad \text{or} \quad \tilde{m}_\mu = \pm\left[\frac{3}{3n-2}\right]^{\frac{1}{2}}$$

These are called *mixture states*, since they correspond to microscopic configurations correlated equally with a finite number n of the stored patterns (or their negatives). Without loss of generality we can always perform transformations on the set of stored patterns (permutations and reflections), such that these mixture states acquire the form

$$\boldsymbol{m} = m_n(\overbrace{1,\ldots,1}^{n \text{ times}}, \overbrace{0,\ldots,0}^{p-n \text{ times}}) \qquad m_n = \left[\frac{3}{3n-2}\right]^{\frac{1}{2}}(\beta-1)^{1/2} + \ldots \tag{3.39}$$

These states (3.39) are indeed solutions of (3.38):

$$\begin{aligned} \mu \leq n: &\qquad m_n = \langle \xi_\mu \tanh\left[\beta m_n \textstyle\sum_{\nu\leq n}\xi_\nu\right]\rangle_{\boldsymbol{\xi}} \\ \mu > n: &\qquad 0 = \langle \xi_\mu \tanh\left[\beta m_n \textstyle\sum_{\nu\leq n}\xi_\nu\right]\rangle_{\boldsymbol{\xi}} \end{aligned}$$

The second equation is automatically satisfied since the average factorises. The first equation leads to a condition determining the amplitude m_n of the mixture states:

$$m_n = \langle \left[\frac{1}{n}\sum_{\mu\leq n}\xi_\mu\right]\tanh\left[\beta m_n \sum_{\nu\leq n}\xi_\nu\right]\rangle_{\boldsymbol{\xi}} \tag{3.40}$$

Stability. The relevant question at this stage is whether these solutions are also stable. Here we will determine the stability for the sequential dynamics case only (for parallel dynamics a similar analysis can be performed, with identical results). Note that the macroscopic dynamic laws, describing the evolution of the overlaps, can itself be written as a gradient descent on a surface $f(\boldsymbol{m})$:

$$\frac{d}{dt}m_\mu = -\frac{\partial}{\partial m_\mu}f(\boldsymbol{m}) \qquad f(\boldsymbol{m}) = \frac{1}{2}\boldsymbol{m}^2 - \frac{1}{\beta}\langle \log\cosh\left[\beta\boldsymbol{\xi}\cdot\boldsymbol{m}\right]\rangle_{\boldsymbol{\xi}} \tag{3.41}$$

Stability of a stationary state is therefore equivalent to this state being a local minimum of $f(\boldsymbol{m})$. The second derivative of $f(\boldsymbol{m})$ is given by

$$\frac{\partial^2 f(\boldsymbol{m})}{\partial m_\mu \partial m_\nu} = \delta_{\mu\nu} - \beta\langle \xi_\mu \xi_\nu \left[1 - \tanh^2\left[\beta \boldsymbol{\xi}\cdot\boldsymbol{m}\right]\right]\rangle_{\boldsymbol{\xi}} \tag{3.42}$$

(a local minimum corresponds to a positive definite second derivative). In the trivial saddle point $\boldsymbol{m} = 0$ this gives simply $\delta_{\mu\nu}[1-\beta]$, so at $T = 1$ this state destabilises. In a mixture state of the type (3.39) the second derivative becomes:

$$D^{(n)}_{\mu\nu} = \delta_{\mu\nu} - \beta\langle \xi_\mu \xi_\nu \left[1 - \tanh^2\left[\beta m_n \sum_{\rho\le n} \xi_\rho\right]\right]\rangle_{\boldsymbol{\xi}} \tag{3.43}$$

Due to the symmetries in the problem the spectrum of the matrix $D^{(n)}$ can be calculated. One finds three distinct eigenspaces:

	Eigenspace :	Eigenvalue :
I :	$\boldsymbol{x} = (0,\ldots,0,x_{n+1},\ldots,x_p)$	$1-\beta[1-Q]$
II :	$\boldsymbol{x} = (1,\ldots,1,0,\ldots,0)$	$1-\beta[1-Q+(1-n)R]$
III :	$\boldsymbol{x} = (x_1,\ldots,x_n,0,\ldots,0),\ \sum_\mu x_\mu = 0$	$1-\beta[1-Q+R]$

with

$$Q = \langle \tanh^2\left[\beta m_n \textstyle\sum_{\rho\le n}\xi_\rho\right]\rangle_{\boldsymbol{\xi}}$$
$$R = \langle \xi_1\xi_2 \tanh^2\left[\beta m_n \textstyle\sum_{\rho\le n}\xi_\rho\right]\rangle_{\boldsymbol{\xi}}$$

Eigenspace *III* and the quantity R only come into play for $n > 1$. To find the smallest eigenvalue we need to know the sign of R. With the abbreviation $M_{\boldsymbol{\xi}} = \sum_{\rho\le n}\xi_\rho$ we find:

$$\begin{aligned} n(n-1)R &= \langle M^2_{\boldsymbol{\xi}} \tanh^2\left[\beta m_n M_{\boldsymbol{\xi}}\right]\rangle_{\boldsymbol{\xi}} - n\langle \tanh^2\left[\beta m_n M_{\boldsymbol{\xi}}\right]\rangle_{\boldsymbol{\xi}} \\ &= \langle M^2_{\boldsymbol{\xi}} \tanh^2\left[\beta m_n M_{\boldsymbol{\xi}}\right]\rangle_{\boldsymbol{\xi}} - \langle M^2_{\boldsymbol{\xi}}\rangle_{\boldsymbol{\xi}}\langle \tanh^2\left[\beta m_n M_{\boldsymbol{\xi}}\right]\rangle_{\boldsymbol{\xi}} \\ &\ge 0 \end{aligned}$$

We may now identify the conditions for an n-mixture state to correspond to a local minimum of $f(\boldsymbol{m})$. For $n = 1$ the relevant eigenvalue is I, now the quantity Q simplifies considerably. For $n > 1$ the relevant eigenvalue is *III*, here we can combine Q and R into one single average (which reduces to a trivial expression for $n = 2$):

$$\begin{aligned} n = 1 :&\quad 1-\beta\left[1-\tanh^2[\beta m_1]\right] > 0 \\ n = 2 :&\quad 1-\beta > 0 \\ n \ge 3 :&\quad 1-\beta\left[1-\langle \tanh^2[\beta m_n \textstyle\sum_{\rho=3}^n \xi_\rho]\rangle_{\boldsymbol{\xi}}\right] > 0 \end{aligned}$$

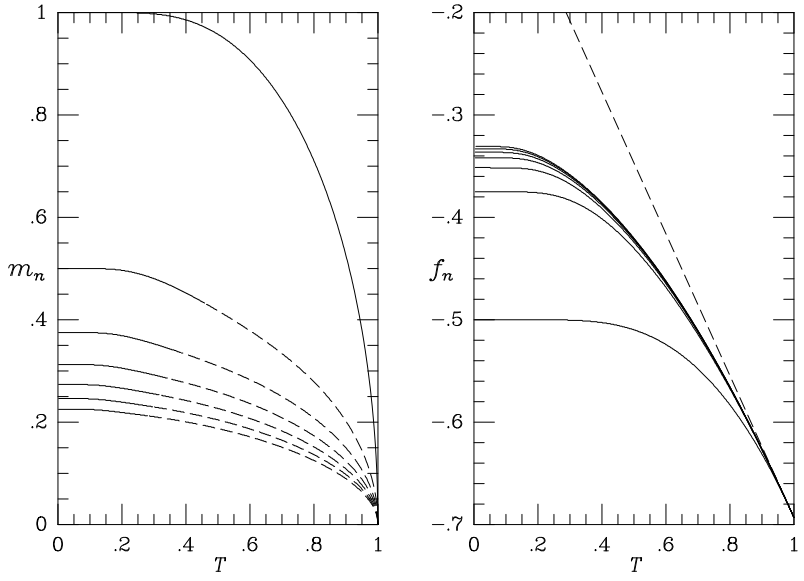


Figure 3.8: Left picture: Amplitudes m_n of the mixture states of the Hopfield model as a function of the noise level T. From top to bottom: $n = 1, 3, 5, 7, 9, 11, 13$. Solid: region where they are stable (i.e. local minima of f). Dashed: region where they are unstable. Right picture: corresponding values f_n. From bottom to top: $n = 1, 3, 5, 7, 9, 11, 13$. Dashed line: value f_0 for the disordered state $m = 0$ (for comparison).

The pure $n = 1$ states, correlated with one pattern only, are the desired solutions. They turn out to be stable for all $T < 1$, since partial differentiation with respect to β of the $n = 1$ amplitude equation (3.40) gives

$$m_1 = \tanh[\beta m_1] \quad \rightarrow \quad 1 - \beta \left[1 - \tanh^2[\beta m_1]\right] = \frac{m_1 \left[1 - \tanh^2[\beta m_1]\right]}{\partial m_1 / \partial \beta} > 0$$

(clearly $\mathrm{sgn}[m_1] = \mathrm{sgn}[\partial m_1 / \partial \beta]$). The $n = 2$ mixtures are always unstable. For $n \geq 3$ we have to solve the amplitude equations (3.40) numerically to evaluate their stability. The result is shown in figure 3.8, together with the corresponding values f_n of the Lyapunov function $f(\boldsymbol{m})$ (3.41). It turns out that only for odd n will there be a critical temperature below which the n-mixture states are local minima of $f(\boldsymbol{m})$. From figure 3.8 we can also conclude that, in terms of the network functioning as an associative memory, noise is actually benificial in the sense that it can be used to eliminate the unwanted $n > 1$ mixture attractors (whilst retaining the relevant ones: the pure $n = 1$ states).

In fact the overlap equations (3.38) do also allow for stable solutions different from the n-mixture states discussed here. They are in turn found to be continuously bifurcating mixtures of the mixture states. However, for random (or uncorrelated) patterns they come into existence only near $T = 0$ and play a marginal role; state space is dominated by the odd n-mixture states.

Appendix A

Conditions for the Central Limit Theorem to Apply

We first give a simple *necessary* condition for a random variable to have a Gaussian probability distribution. Lindeberg's Theorem gives a *sufficient* condition for an infinite sum of independent random variables to have a Gaussian probability distribution. We apply both to expressions of the form $\sum_{i=1}^{N} W_i x_i$, in which the $x_i \in \{-1, 1\}$ are independent zero average random variables.

Moment Condition

Consider a zero average random variable Y, described by a Gaussian probability distribution

$$P(Y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}Y^2/\sigma^2}$$

All odd moments are zero, i.e. $\langle Y^{2m+1} \rangle = 0$, due to the symmetry $P(Y) = P(-Y)$. All even moments $\langle Y^{2m} \rangle$ can be expressed in terms of σ. For instance

$$\langle Y^2 \rangle = \int \frac{dY}{\sigma\sqrt{2\pi}} Y^2 e^{-\frac{1}{2}Y^2/\sigma^2} = [2\pi\sigma^2]^{-\frac{1}{2}} \lim_{x \to 1/2\sigma^2} \int dY \; Y^2 e^{-xY^2}$$

$$= -[2\pi\sigma^2]^{-\frac{1}{2}} \lim_{x \to 1/2\sigma^2} \frac{d}{dx} \int dY \; e^{-xY^2} = -[2\pi\sigma^2]^{-\frac{1}{2}} \lim_{x \to 1/2\sigma^2} \frac{d}{dx} \sqrt{\pi} x^{-\frac{1}{2}} = \sigma$$

(where we used $\int dy \; e^{-\frac{1}{2}y^2} = \sqrt{2\pi}$, see Appendix B). We obtain in a similar way:

$$\langle Y^4 \rangle = \int \frac{dY}{\sigma\sqrt{2\pi}} Y^4 e^{-\frac{1}{2}Y^2/\sigma^2} = [2\pi\sigma^2]^{-\frac{1}{2}} \lim_{x \to 1/2\sigma^2} \int dY \; Y^4 e^{-xY^2}$$

$$= [2\pi\sigma^2]^{-\frac{1}{2}} \lim_{x \to 1/2\sigma^2} \frac{d^2}{dx^2} \int dY \; e^{-xY^2} = [2\pi\sigma^2]^{-\frac{1}{2}} \lim_{x \to 1/2\sigma^2} \frac{d^2}{dx^2} \sqrt{\pi} x^{-\frac{1}{2}} = 3\sigma^2$$

A necessary condition for Y to have a Gaussian probability distribution is thus $\langle Y^4 \rangle = 3\langle Y^2 \rangle^2$.

We now choose $Y_N = \sum_{i=1}^{N} W_i x_i [\sum_{i=j}^{N} W_j^2]^{-\frac{1}{2}}$, where the $x_k \in \{-1, 1\}$ are independent random variables with $\langle x_k \rangle = 0$. A necessary condition for Y_N to have a Gaussian probability

distribution for $N \to \infty$ is apparently

$$\lim_{N\to\infty} \frac{\sum_{ijkl=1}^{n} W_i W_j W_k W_l \langle x_i x_j x_k x_l \rangle}{\left[\sum_{i=1}^{N} W_i^2\right]^2} = 3$$

With the help of the identity $\langle x_i x_j x_k x_l \rangle = \delta_{ij}\delta_{kl} + \delta_{ik}\delta_{jl} + \delta_{il}\delta_{jk} - 2\delta_{ij}\delta_{kl}\delta_{ik}$ this gives

$$\lim_{N\to\infty} \frac{\sum_{i=1}^{N} W_i^4}{[\sum_{j=1}^{N} W_j^2]^2} = 0 \tag{A.1}$$

Lindeberg's Theorem

Let $X_1, X_2, X_3, \ldots$ be a sequence of independent random variables such that $\langle X_k \rangle = 0$ and $\langle X_k^2 \rangle = \sigma_k^2$. Let $p_k(x)$ denote the distribution function of X_k and put $S_n = \sum_{i=1}^{n} X_i$, so that

$$\langle S_n \rangle = 0 \qquad s_n^2 = \langle S_n^2 \rangle = \sigma_1^2 + \sigma_2^2 + \sigma_3^2 + \cdots + \sigma_n^2$$

Suppose that the Lindeberg condition is satisfied i.e.

$$\text{for each } t > 0: \qquad \lim_{n\to\infty} \frac{1}{s_n^2} \sum_{k=1}^{n} \int_{|x| \geq t s_n} dx\ x^2 p_k(x) \ = \ 0 \tag{A.2}$$

Then the distribution of the normalised sum S_n/s_n tends to the zero average and unit variance Gaussian distribution as $n \to \infty$. For the proof of this theorem see W. Feller, 1966, 'An Introduction to Probability and its Applications II', Wiley, New York.

We now choose $X_i = W_i x_i$, where the $x_k \in \{-1, 1\}$ are independent random variables with $\langle x_k \rangle = 0$. Thus $\sigma_k^2 = W_k^2$ for each k. Lindeberg's condition (A.2) now reads:

$$\text{for each } t > 0: \qquad \lim_{N\to\infty} \frac{\sum_{i=1}^{N} W_i^2 \theta[|W_i| - t\sqrt{\sum_{j=1}^{N} W_j^2}]}{\sum_{i=1}^{N} W_i^2} \ = \ 0$$

We can simplify this condition further. Upon defining $v_k = W_k / \sqrt{\sum_{j=1}^{N} W_j^2}$ our condition for asymptotically Gaussian behaviour can be written as

$$\text{for each } \epsilon > 0: \qquad \lim_{N\to\infty} \sum_{i=1}^{N} v_i^2 \theta[v_i^2 - \epsilon] = 0$$

Since all non-zero terms in this sum obey ${v_k}^2 \geq \epsilon$, and since ${v_k}^2 \leq 1$, it follows that

$$\epsilon \sum_{k=1}^{N} \theta[v_k^2 - \epsilon] \leq \sum_{k=1}^{N} v_k^2 \theta[v_k^2 - \epsilon] \leq \sum_{k=1}^{N} \theta[v_k^2 - \epsilon]$$

This tells us that here the Lindeberg condition is equivalent to:

$$\text{for each } \epsilon > 0: \qquad \lim_{N\to\infty} \sum_{i=1}^{N} \theta[v_i^2 - \epsilon] = 0$$

In terms of the original variables W_i this reads:

$$\text{for each } \epsilon > 0: \qquad \lim_{N\to\infty} \sum_{i=1}^{N} \theta\left[W_i^2 - \epsilon \sum_{k=1}^{N} W_k^2\right] = 0 \tag{A.3}$$

Appendix B

Gaussian Integrals

In this appendix we derive some properties of symmetric positive definite matrices $\boldsymbol{A}$, and their associated Gaussian integrals in $\Re^N$:

$$I = \int d\boldsymbol{x}\ f(\boldsymbol{x}) e^{-\frac{1}{2}\boldsymbol{x}\cdot\boldsymbol{A}\boldsymbol{x}}$$

(for simple functions f), as well as calculate explicitly some integrals of this form, with specific choices of the matrix $\boldsymbol{A}$, that we encounter in the lectures.

Real, Symmetric, Positive Definite Matrices. The symmetric $N \times N$ matrix $\boldsymbol{A}$ is assumed to be positive definite, i.e. $\boldsymbol{x} \cdot \boldsymbol{A}\boldsymbol{x} > 0$ for all $\boldsymbol{x} \in \Re^N$ with $|\boldsymbol{x}| \neq 0$. The eigenvalue polynomial $\det[\boldsymbol{A} - \lambda \mathbb{1}] = 0$ is of order N, so $\boldsymbol{A}$ will have N (possibly complex) solutions λ (some may coincide) of the eigenvalue problem

$$\boldsymbol{A}\boldsymbol{x} = \lambda \boldsymbol{x}, \qquad \boldsymbol{x} \neq 0 \tag{B.1}$$

We denote complex conjugation of complex numbers z in the usual way: $z = a + ib$, $z^* = a - ib$ $(a, b \in \Re)$, and $|z|^2 = z^* z \in \Re$. We denote the unit matrix in $\Re^N$ with $\mathbb{1}$, so $\mathbb{1}_{ij} = \delta_{ij}$.

Fact 1: All eigenvalues of the matrix $\boldsymbol{A}$ are real.

Proof: In (B.1) we take the inner product with the conjugate vector $\boldsymbol{x}^*$, which gives

$$\sum_{i,j=1}^{N} x_i^* A_{ij} x_j = \lambda \sum_{i=1}^{N} |x_i|^2$$

We use the symmetry of $\boldsymbol{A}$, and substitute $A_{ij} \to \frac{1}{2}[A_{ij} + A_{ji}]$:

$$\lambda = \frac{1}{2} \frac{\sum_{ij} x_i^* [A_{ij} + A_{ji}] x_j}{\sum_{i=1}^{N} |x_i|^2} = \frac{1}{2} \frac{\sum_{ij} A_{ij} [x_i^* x_j + x_i x_j^*]}{\sum_{i=1}^{N} |x_i|^2}$$

Since $[x_i^* x_j + x_i x_j^*]^* = x_i x_j^* + x_i^* x_j = x_i^* x_j + x_i x_j^*$, the above fraction is entirely real-valued, so $\lambda \in \Re$.

Fact 2: All eigenvectors can be chosen real-valued.

Proof: For a given eigenvalue λ the corresponding eigenvectors $\boldsymbol{x}$ are the solutions of (B.1). We separate real and imaginary parts of every eigenvector:

$$\boldsymbol{x} = \mathrm{Re}\boldsymbol{x} + i\mathrm{Im}\boldsymbol{x} \qquad \mathrm{Re}\boldsymbol{x} = \frac{1}{2}[\boldsymbol{x} + \boldsymbol{x}^*] \qquad \mathrm{Im}\boldsymbol{x} = \frac{1}{2i}[\boldsymbol{x} - \boldsymbol{x}^*]$$

with $\mathrm{Re}\boldsymbol{x} \in \Re^N$ and $\mathrm{Im}\boldsymbol{x} \in \Re^N$. Taking the complex conjugate of equation (B.1) gives $\boldsymbol{A}\boldsymbol{x}^* = \lambda\boldsymbol{x}^*$ (since λ is real). Apparently, if $\boldsymbol{x}$ is an eigenvector with eigenvalue λ, so is $\boldsymbol{x}^*$. By adding/subtracting the conjugate equation to/from the original equation (B.1) it follows, in turn: if $\boldsymbol{x}$ and $\boldsymbol{x}^*$ are eigenvectors, so are $\mathrm{Re}\boldsymbol{x}$ and $\mathrm{Im}\boldsymbol{x}$. Complex eigenvectors always come in conjugate pairs, and, since the space spanned by $\boldsymbol{x}$ and $\boldsymbol{x}^*$ is the same as the space spanned by $\mathrm{Re}\boldsymbol{x}$ and $\mathrm{Im}\boldsymbol{x}$, we are always allowed to choose the equivalent real-valued pair $\mathrm{Re}\boldsymbol{x}$ and $\mathrm{Im}\boldsymbol{x}$.

Fact 3: All eigenvalues λ are positive.

Proof: From the eigenvalue equation (B.1) we derive this property by taking the inner product with $\boldsymbol{x}$: $\lambda = (\boldsymbol{x} \cdot \boldsymbol{A}\boldsymbol{x})/(\boldsymbol{x}^2) > 0$, since $\boldsymbol{A}$ is positive definite and $\boldsymbol{x}$ is real and nonzero.

Fact 4: For every linear subspace $L \subseteq \Re^N$ the following holds:

$$\text{if} \quad \boldsymbol{A}L \subseteq L \quad \text{then also} \quad \boldsymbol{A}L^\perp \subseteq L^\perp$$

in which $L^\perp$ denotes the orthogonal complement, i.e. $\Re^N = L \otimes L^\perp$.

Proof: For each $\boldsymbol{x} \in L$ we find $(\boldsymbol{x} \cdot \boldsymbol{A}\boldsymbol{y}) = (\boldsymbol{y} \cdot \boldsymbol{A}\boldsymbol{x}) = 0$ (since $\boldsymbol{A}\boldsymbol{x} \in L$ and $\boldsymbol{y} \in L^\perp$). Therefore $\boldsymbol{A}\boldsymbol{y} \in L^\perp$, which completes the proof.

Fact 5: We can construct a complete orthogonal basis in $\Re^N$ of $\boldsymbol{A}$-eigenvectors.

Proof: Consider two eigenvectors $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ of $\boldsymbol{A}$, corresponding to different eigenvalues:

$$\boldsymbol{A}\boldsymbol{x}_a = \lambda_a\boldsymbol{x}_a \qquad \boldsymbol{A}\boldsymbol{x}_b = \lambda_b\boldsymbol{x}_b \qquad \lambda_a \neq \lambda_b$$

We form:

$$0 = (\boldsymbol{x}_a \cdot \boldsymbol{A}\boldsymbol{x}_b) - (\boldsymbol{x}_a \cdot \boldsymbol{A}\boldsymbol{x}_b) = (\boldsymbol{x}_a \cdot \boldsymbol{A}\boldsymbol{x}_b) - (\boldsymbol{x}_b \cdot \boldsymbol{A}\boldsymbol{x}_a)$$
$$= \lambda_b(\boldsymbol{x}_a \cdot \boldsymbol{x}_b) - \lambda_a(\boldsymbol{x}_b \cdot \boldsymbol{x}_a) = (\lambda_a - \lambda_b)(\boldsymbol{x}_a \cdot \boldsymbol{x}_b)$$

Since $\lambda_a \neq \lambda_b$ it follows that $\boldsymbol{x}_a \cdot \boldsymbol{x}_b = 0$. Eigenspaces corresponding to different eigenvalues are mutually orthogonal. If all eigenvalues are distinct, this completes the proof, there being N eigenvalues with corresponding eigenvectors $\boldsymbol{x} \neq 0$. Since these

are proven orthogonal, after normalisation $\boldsymbol{x} \to \boldsymbol{x}/|\boldsymbol{x}|$ they form a complete orthogonal basis.

To deal with degenerate eigenvalues we need Fact 4. For every symmetric $N \times N$ matrix we know: if $\boldsymbol{A}\boldsymbol{x} = \lambda\boldsymbol{x}$, then $\forall \boldsymbol{y}$ with $\boldsymbol{x} \cdot \boldsymbol{y} = 0$: $(\boldsymbol{A}\boldsymbol{y}) \cdot \boldsymbol{x} = 0$. Having found such an eigenvector for a given eigenvalue λ (not unique in the case of a degenerate eigenvalue), a new reduced $(N-1) \times (N-1)$ matrix can be constructed by restricting ourselves to the subspace $\boldsymbol{x}^{\perp}$. The new matrix is again symmetric, the eigenvalue polynomial is of order $N-1$ (and contains all the previous roots except for one corresponding to the eigenvector just eliminated), and we can repeat the argument. This shows that there *must* be N orthogonal eigenvectors, which we can normalise and use as a basis in $\Re^N$.

Final result: there exist a set of vectors $\{\hat{\boldsymbol{e}}^i\}$ $(\lambda = 1, \ldots, N)$ with the properties:

$$\boldsymbol{A}\hat{\boldsymbol{e}}^i = \lambda_i \hat{\boldsymbol{e}}^i \qquad \lambda_i \in \Re,\ \lambda_i > 0 \qquad \hat{\boldsymbol{e}}^i \in \Re^N,\ \hat{\boldsymbol{e}}^i \cdot \hat{\boldsymbol{e}}^j = \delta_{ij} \tag{B.2}$$

We can now bring $\boldsymbol{A}$ onto diagonal form by a simple unitary transformation $\boldsymbol{U}$, which we construct from the components of the normalised eigenvectors $\hat{\boldsymbol{e}}$: $U_{ij} = \hat{e}_i^j$. We denote the transpose of $\boldsymbol{U}$ by $\boldsymbol{U}^\dagger$, $U^\dagger_{ij} = U_{ji}$, and show that $\boldsymbol{U}$ is indeed unitary, i.e. $\boldsymbol{U}^\dagger\boldsymbol{U} = \boldsymbol{U}\boldsymbol{U}^\dagger = \mathbb{1}$:

$$\sum_j (\boldsymbol{U}^\dagger\boldsymbol{U})_{ij} x_j = \sum_{jk} U^{ki} U_{kj} x_j = \sum_{jk} \hat{e}^i_k \hat{e}^j_k x_j = \sum_j \delta_{ij} x_j = x_i$$

$$\sum_j (\boldsymbol{U}\boldsymbol{U}^\dagger)_{ij} x_j = \sum_{jk} U^{ik} U_{jk} x_j = \sum_{jk} \hat{e}^k_i \hat{e}^k_j x_j = \sum_k \hat{e}^k_i (\hat{\boldsymbol{e}} \cdot \boldsymbol{x}) = x_i$$

(since $\{\hat{\boldsymbol{e}}^\ell\}$ forms a complete orthogonal basis). From $\boldsymbol{U}$ being unitary it follows that $\boldsymbol{U}$ and $\boldsymbol{U}^\dagger$ leave inner products, and therefore also lengths, invariant:

$$\boldsymbol{U}\boldsymbol{x} \cdot \boldsymbol{U}\boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{U}^\dagger\boldsymbol{U}\boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{y} \qquad \boldsymbol{U}^\dagger\boldsymbol{x} \cdot \boldsymbol{U}^\dagger\boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{U}\boldsymbol{U}^\dagger\boldsymbol{y} = \boldsymbol{x} \cdot \boldsymbol{y}$$

We can see explicitly that $\boldsymbol{U}$ indeed brings $\boldsymbol{A}$ onto diagonal form:

$$(\boldsymbol{U}^\dagger\boldsymbol{A}\boldsymbol{U})_{ij} = \sum_{kl=1}^{N} U^\dagger_{ik} A_{kl} U_{lj} = \sum_{kl=1}^{N} \hat{e}^i_k A_{kl} \hat{e}^j_l = \lambda_j \sum_{k=1}^{N} \hat{e}^i_k \hat{e}^j_k = \lambda_j \delta_{ij} \tag{B.3}$$

Note that the inverse $\boldsymbol{A}^{-1}$ of the matrix $\boldsymbol{A}$ exists, and can be written as follows:

$$(A^{-1})_{ij} = \sum_{k=1}^{N} \lambda_k^{-1} \hat{e}^k_i \hat{e}^k_j \tag{B.4}$$

To prove that this is indeed the inverse of $\boldsymbol{A}$, we just work out for any $\boldsymbol{x} \in \Re^N$ the two expressions

$$(\boldsymbol{A}\boldsymbol{A}^{-1}\boldsymbol{x})_i = \sum_{kj=1}^{N} A_{ik} \sum_{\ell=1}^{N} \lambda_\ell^{-1} \hat{e}^\ell_k \hat{e}^\ell_j x_j = \sum_{\ell=1}^{N} \hat{e}^\ell_i (\hat{\boldsymbol{e}}^\ell \cdot \boldsymbol{x}) = x_i$$

(again since $\{\hat{\boldsymbol{e}}^\ell\}$ forms a complete orthogonal basis), and

$$(\boldsymbol{A}^{-1}\boldsymbol{A}\boldsymbol{x})_i = \sum_{kj=1}^{N} \sum_{\ell=1}^{N} \lambda_\ell^{-1} \hat{e}^\ell_i \hat{e}^\ell_k A_{kj} x_j = \sum_{\ell=1}^{N} \hat{e}^\ell_i (\hat{\boldsymbol{e}}^\ell \cdot \boldsymbol{x}) = x_i$$

Gaussian Integrals. We now turn to the associated Gaussian integrals

$$I = \int d\boldsymbol{x}\ f(\boldsymbol{x}) e^{-\frac{1}{2}\boldsymbol{x}\cdot\boldsymbol{A}\boldsymbol{x}} \tag{B.5}$$

The simplest such integral is

$$\int dx\ e^{-\frac{1}{2}x^2} = \sqrt{2\pi}$$

(for a proof see the last part of this appendix). For $f(\boldsymbol{x}) = 1$ we can do the integral (B.5) by using the previous results on the diagonalisability of the matrix $\boldsymbol{A}$. We put $\boldsymbol{x} = \boldsymbol{U}\boldsymbol{z}$ (since $\boldsymbol{U}$ leaves inner products invariant: $d\boldsymbol{x} = d\boldsymbol{z}$):

$$\int d\boldsymbol{x}\ e^{-\frac{1}{2}\boldsymbol{x}\cdot\boldsymbol{A}\boldsymbol{x}} = \int d\boldsymbol{z}\ e^{-\frac{1}{2}\boldsymbol{z}\cdot\boldsymbol{U}^\dagger\boldsymbol{A}\boldsymbol{U}\boldsymbol{z}} = \prod_{\ell=1}^{N}\left[\int dz\ e^{-\frac{1}{2}\lambda_\ell z^2}\right]$$

$$= \left[\prod_{\ell=1}^{N}\frac{1}{\sqrt{\lambda_\ell}}\right]\left[\int dz\ e^{-\frac{1}{2}z^2}\right]^N = \frac{(2\pi)^{N/2}}{\sqrt{\det\boldsymbol{A}}} \tag{B.6}$$

(note: the determinant of $\boldsymbol{A}$ is unvariant under rotations, so it can be evaluated with $\boldsymbol{A}$ on diagonal form, which gives the product of the N eigenvalues).

Due to the symmetry of the integrand in (B.5) under reflection $\boldsymbol{x} \to -\boldsymbol{x}$, the integral reduces to zero for $f(\boldsymbol{x}) = x_i$. For $f(\boldsymbol{x}) = x_i x_j$ we find:

$$\int d\boldsymbol{x}\ x_i x_j e^{-\frac{1}{2}\boldsymbol{x}\cdot\boldsymbol{A}\boldsymbol{x}} = \lim_{\boldsymbol{b}\to 0}\frac{\partial^2}{\partial b_i \partial b_j}\int d\boldsymbol{x}\ e^{-\frac{1}{2}\boldsymbol{x}\cdot\boldsymbol{A}\boldsymbol{x}+\boldsymbol{b}\cdot\boldsymbol{x}}$$

$$= \lim_{\boldsymbol{b}\to 0}\frac{\partial^2}{\partial b_i \partial b_j}\int d\boldsymbol{z}\ e^{-\frac{1}{2}\boldsymbol{z}\cdot\boldsymbol{U}^\dagger\boldsymbol{A}\boldsymbol{U}\boldsymbol{z}+\boldsymbol{z}\cdot\boldsymbol{U}^\dagger\boldsymbol{b}}$$

$$= \lim_{\boldsymbol{b}\to 0}\frac{\partial^2}{\partial b_i \partial b_j}\prod_{\ell=1}^{N}\left[\int dz\ e^{-\frac{1}{2}\lambda_\ell z^2 + z(\boldsymbol{U}^\dagger\boldsymbol{b})_\ell}\right]$$

$$= \lim_{\boldsymbol{b}\to 0}\frac{\partial^2}{\partial b_i \partial b_j}\prod_{\ell=1}^{N}\left[\int dz\ e^{-\frac{1}{2}\lambda_\ell[z-(\boldsymbol{U}^\dagger\boldsymbol{b})_\ell\lambda_\ell^{-1}]^2+\frac{1}{2}\lambda_\ell^{-1}(\boldsymbol{U}^\dagger\boldsymbol{b})_\ell^2}\right]$$

$$= \lim_{\boldsymbol{b}\to 0}\frac{\partial^2}{\partial b_i \partial b_j} e^{\frac{1}{2}\sum_{ij\ell=1}^{N}\lambda_\ell^{-1}U_{i\ell}b_i U_{j\ell}b_j}\prod_{\ell=1}^{N}\left[\int dz\ e^{-\frac{1}{2}\lambda_\ell z^2}\right]$$

$$= \frac{(2\pi)^{N/2}}{\sqrt{\det\boldsymbol{A}}}\lim_{\boldsymbol{b}\to 0}\frac{\partial^2}{\partial b_i \partial b_j} e^{\frac{1}{2}\sum_{ij=1}^{N} b_i b_j \sum_{\ell=1}^{N}\lambda_\ell^{-1}\hat{e}_i^\ell\hat{e}_j^\ell}$$

$$= (A^{-1})_{ij}\frac{(2\pi)^{N/2}}{\sqrt{\det\boldsymbol{A}}} \tag{B.7}$$

In particular, by combining the last two results, we find a powerful (and completely general) relation for Gaussian probability distributions with zero mean $\boldsymbol{x} = 0$ (the latter one can always achieve by a simple translation):

$$\frac{\int d\boldsymbol{x}\ x_i x_j e^{-\frac{1}{2}\boldsymbol{x}\cdot\boldsymbol{A}\boldsymbol{x}}}{\int d\boldsymbol{x}\ e^{-\frac{1}{2}\boldsymbol{x}\cdot\boldsymbol{A}\boldsymbol{x}}} = (A^{-1})_{ij} \tag{B.8}$$

If we know that a given distribution is Gaussian, with zero average, we apparently only need to calculate the correlations $\langle x_i x_j \rangle$ to know the full distribution:

$$P(\boldsymbol{x}) \text{ Gaussian, with } \langle \boldsymbol{x} \rangle = 0 \qquad \Rightarrow \qquad P(\boldsymbol{x}) = \frac{e^{-\frac{1}{2}\boldsymbol{x}\cdot\boldsymbol{A}\boldsymbol{x}}}{(2\pi)^{N/2}\det{}^{-\frac{1}{2}}\boldsymbol{A}}, \text{ with } (A^{-1})_{ij} = \langle x_i x_j \rangle \tag{B.9}$$

Specific Integrals. Finally we calculate explicitly the Gaussian integrals we encountered in the lectures.

Integral 1:

$$I = \int dz \; e^{-\frac{1}{2}z^2} = \sqrt{2\pi} \tag{B.10}$$

Proof: Write the square of the integral as a single integral in $\Re^2$, and switch to polar coordinates, ($z_1 = r\cos\phi, \; z_2 = r\sin\phi$). The Jacobian of this coordinate transformation is simply r.

$$I^2 = \int dz_1 dz_2 \; e^{-\frac{1}{2}\boldsymbol{z}^2} = \int_0^{2\pi} d\phi \int_0^\infty dr \; r e^{-\frac{1}{2}r^2} = 2\pi \left[-e^{-\frac{1}{2}r^2} \right]_0^\infty = 2\pi$$

Therefore $I = \sqrt{2\pi}$.

Integral 2:

$$I = \int_0^\infty \int_0^\infty \frac{dudv}{\pi\sqrt{1-\omega^2}} v e^{-\frac{1}{2}[u^2+v^2+2\omega uv]/(1-\omega^2)} = \frac{1-\omega}{\sqrt{2\pi}} \tag{B.11}$$

Proof:

$$I = \frac{1-\omega^2}{\pi} \int_0^\infty \int_0^\infty dudv \; v e^{-\frac{1}{2}[u^2+v^2+2\omega uv]}$$

$$= \frac{1-\omega^2}{\pi} \int_0^\infty dv \; v e^{-\frac{1}{2}v^2} \int_0^\infty du \; e^{-\frac{1}{2}[u+\omega v]^2 + \frac{1}{2}\omega^2 v^2}$$

$$= \frac{1}{\pi} \int_0^\infty dv \; v e^{-\frac{1}{2}v^2} \int_{\omega v/\sqrt{1-\omega^2}}^\infty du \; e^{-\frac{1}{2}u^2}$$

$$= -\frac{1}{\pi} \int_0^\infty dv \left\{ \frac{\partial}{\partial v} e^{-\frac{1}{2}v^2} \right\} \int_{\omega v/\sqrt{1-\omega^2}}^\infty du \; e^{-\frac{1}{2}u^2}$$

$$= -\frac{1}{\pi} \left[e^{-\frac{1}{2}v^2} \int_{\omega v/\sqrt{1-\omega^2}}^\infty du \; e^{-\frac{1}{2}u^2} \right]_0^\infty + \frac{1}{\pi} \int_0^\infty dv \; e^{-\frac{1}{2}v^2} \frac{\partial}{\partial v} \int_{\omega v/\sqrt{1-\omega^2}}^\infty du \; e^{-\frac{1}{2}u^2}$$

$$= \frac{1}{\sqrt{2\pi}} - \frac{\omega}{\pi\sqrt{1-\omega}} \int_0^\infty dv \; e^{-\frac{1}{2}v^2 - \frac{1}{2}v^2\omega^2/(1-\omega^2)}$$

$$= \frac{1}{\sqrt{2\pi}} - \frac{\omega}{\pi\sqrt{1-\omega}} \int_0^\infty dv \; e^{-\frac{1}{2}v^2/(1-\omega^2)}$$

$$= \frac{1}{\sqrt{2\pi}} - \frac{\omega}{\sqrt{2\pi}} = \frac{1-\omega}{\sqrt{2\pi}}$$

Integral 3:

$$I = \int_0^\infty\int_0^\infty \frac{dudv}{\pi\sqrt{1-\omega^2}} e^{-\frac{1}{2}[u^2+v^2+2\omega uv]/(1-\omega^2)} = \frac{1}{\pi}\arccos(\omega) \tag{B.12}$$

Proof:

$$I = \frac{\sqrt{1-\omega^2}}{\pi}\int_0^\infty\int_0^\infty dudv\ e^{-\frac{1}{2}[u^2+v^2+2\omega uv]}$$

We introduce polar coordinates ($u = r\cos\phi, v = r\sin\phi$):

$$I = \frac{\sqrt{1-\omega^2}}{\pi}\int_0^{\pi/2} d\phi \int_0^\infty dr\ re^{-\frac{1}{2}r^2[1+\omega\sin(2\phi)]}$$

$$= \frac{\sqrt{1-\omega^2}}{2\pi}\int_0^{\pi} \frac{d\phi}{1+\omega\sin(\phi)} \int_0^\infty dr\ re^{-\frac{1}{2}r^2}$$

$$= \frac{\sqrt{1-\omega^2}}{2\pi}\int_0^{\pi} \frac{d\phi}{1+\omega\sin(\phi)} \left[-re^{-\frac{1}{2}r^2}\right]_0^\infty$$

$$= \frac{\sqrt{1-\omega^2}}{2\pi}\int_0^{\pi} \frac{d\phi}{1+\omega\sin(\phi)}$$

The ϕ integral can be found in I.S. Gradshteyn and I.M. Ryzhik (1980), 'Table of Integrals, Series and Products', Academic Press, London:

$$I = \frac{\sqrt{1-\omega^2}}{2\pi}\left[\frac{2}{\sqrt{1-\omega^2}}\arctan\left(\frac{\omega+\tan(\frac{1}{2}\phi)}{\sqrt{1-\omega^2}}\right)\right]_0^\pi$$

$$= \frac{1}{\pi}\left\{\frac{\pi}{2} - \arctan\left(\frac{\omega}{\sqrt{1-\omega^2}}\right)\right\}$$

Finally, using $\cos[\frac{\pi}{2}-\psi] = \sin\psi$, we find

$$I = \frac{1}{\pi}\arccos(\omega)$$

Appendix C

The δ-Distribution

Definition. There are several ways of introducing the δ-distribution. Here we will go for an intuitive definition first, and a formal one later. We define the δ-distribution as the probability distribution $\delta(x)$ corresponding to a random variable in the limit where the randomness in the variable vanishes. If x is 'distributed' around zero, this implies

$$\int dx\ f(x)\delta(x) = f(0) \qquad \text{for any function } f$$

The problem arises when we want to actually write down an expression for $\delta(x)$. Intuitively one could think of writing something like

$$\delta(x) = \lim_{\Delta\to 0} G_\Delta(x) \qquad G_\Delta(x) = \frac{1}{\Delta\sqrt{2\pi}} e^{-\frac{1}{2}x^2/\Delta^2} \tag{C.1}$$

This is not a true function in a mathematical sense; $\delta(x)$ is zero for $x \neq 0$ and $\delta(0) = \infty$. The way to interpret and use expressions like (C.1) is to realise that $\delta(x)$ only has a meaning when appearing inside an integration. One then takes the limit $\Delta \to 0$ *after* performing the integration. Upon adopting this convention, we can use (C.1) to derive the following properties (for sufficiently well-behaved and differentiable functions f[1]):

$$\int dx\ \delta(x)f(x) = \lim_{\Delta\to 0}\int dx\ G_\Delta(x)f(x) = \lim_{\Delta\to 0}\int \frac{dx}{\sqrt{2\pi}}\ e^{-\frac{1}{2}x^2} f(\Delta x) = f(0)$$

$$\int dx\ \delta'(x)f(x) = \lim_{\Delta\to 0}\int dx \left\{ \frac{d}{dx}\left[G_\Delta(x)f(x)\right] - G_\Delta(x)f'(x)\right\}$$

$$= \lim_{\Delta\to 0}\left[G_\Delta(x)f(x)\right]_{-\infty}^{\infty} - f'(0) = -f'(0)$$

both can be summarised in and generalised to the single expression:

$$\int dx\ f(x)\frac{d^n}{dx^n}\delta(x) = (-1)^n \lim_{x\to 0}\frac{d^n}{dx^n}f(x) \qquad (n = 0, 1, 2, \ldots) \tag{C.2}$$

[1] The conditions on the so-called 'test-functions' f can be properly formalised; this being not a course on distribution theory, here we just concentrate on the basic ideas and properties

Equivalently we can take the result (C.2) as our definition of the δ-distribution.

$\delta(x)$ as Solution of the Liouville Equation. Here we prove that the δ-distribution can be used to represent the solution of the so-called Liouville equation:

$$\frac{\partial}{\partial t}P_t(x) = -\frac{\partial}{\partial x}\left[P_t(x)F(x)\right] \tag{C.3}$$

The general solution of (C.3) is

$$P_t(x) = \int dx_0 \; P_0(x_0)\delta[x - x^\star(t;x_0)] \tag{C.4}$$

in which $x^\star(t;x_0)$ is the solution of the ordinary differential equation

$$\frac{d}{dt}x^\star(t) = F(x^\star(t)) \qquad x^\star(0) = x_0 \tag{C.5}$$

In particular, if $P_0(x_0)$ is a δ-distribution in x_0, the general solution will remain a δ-distribution in x for all times: $P_t(x) = \delta[x - x^\star(t;x_0)]$. The proof that (C.4) is true consists of showing that both sides of (C.3) give the same result inside integrals, if we insert the proposed solution (C.4):

$$\begin{aligned}
&\int dx \; f(x)\left\{\frac{\partial}{\partial t}P_t(x) + \frac{\partial}{\partial x}\left[P_t(x)F(x)\right]\right\} \\
&\quad = \frac{\partial}{\partial t}\int dx \; f(x)P_t(x) + \left[f(x)P_t(x)F(x)\right]_{-\infty}^{\infty} - \int dx \; P_t(x)F(x)f'(x) \\
&\quad = \int dx_0 \; P_0(x_0)\left\{\frac{\partial}{\partial t}f(x^\star(t;x_0)) - F(x^\star(t;x_0))f'(x^\star(t;x_0))\right\} \\
&\quad = \int dx_0 \; P_0(x_0)f'(x^\star(t;x_0))\left\{\frac{d}{dt}x^\star(t;x_0) - F(x^\star(t;x_0))\right\} = 0
\end{aligned}$$

Representations, Relations, Generalisations. We can use the definitions of Fourier transforms and inverse Fourier transforms to obtain an integral representation of the δ-distribution:

$$\mathcal{F}: \; f(x) \to \hat{f}(k) \qquad \hat{f}(k) = \int dx \; e^{-2\pi ikx}f(x)$$

$$\mathcal{F}^{-1}: \; \hat{f}(k) \to f(x) \qquad f(x) = \int dk \; e^{2\pi ikx}\hat{f}(k)$$

In combination these relations give the identity:

$$f(x) = \int dk \; e^{2\pi ikx}\int dy \; e^{-2\pi iky}f(y)$$

Application to $f(x) = \delta(x)$ gives:

$$\delta(x) = \int dk \; e^{2\pi ikx} = \int \frac{dk}{2\pi} \; e^{ikx} \tag{C.6}$$

Another useful relation is the following one, which relates the δ-distribution to the step-function:

$$\delta(x) = \frac{d}{dx}\theta(x) \tag{C.7}$$

This we prove by showing that both have the same effect inside an integration (with an arbitrary test-function):

$$\int dx \left[\delta(x) - \frac{d}{dx}\theta(x)\right] f(x) = f(0) - \lim_{\epsilon\to 0}\int_{-\epsilon}^{\epsilon} dx \left\{\frac{d}{dx}\left[\theta(x)f(x)\right] - f'(x)\theta(x)\right\}$$

$$= f(0) - \lim_{\epsilon\to 0}\left[f(\epsilon) - 0\right] + \lim_{\epsilon\to 0}\int_{0}^{\epsilon} dx\ f'(x) = 0$$

Finally, the following generalisation is straightforward:

$$\boldsymbol{x} \in \mathcal{R}^N : \qquad \delta(\boldsymbol{x}) = \prod_{i=1}^{N} \delta(x_i) \tag{C.8}$$

Appendix D

Exercises

1. Higher Order Synapses

It is known that there also exist synapses which operate in a more complicated way than the simple ones discussed so far. We now try to see how our equations would change if we were to take into account so-called higher-order synapses, like the one in the drawing: The new type of synapse requires the simultaneous arrival of *two* action potentials to release

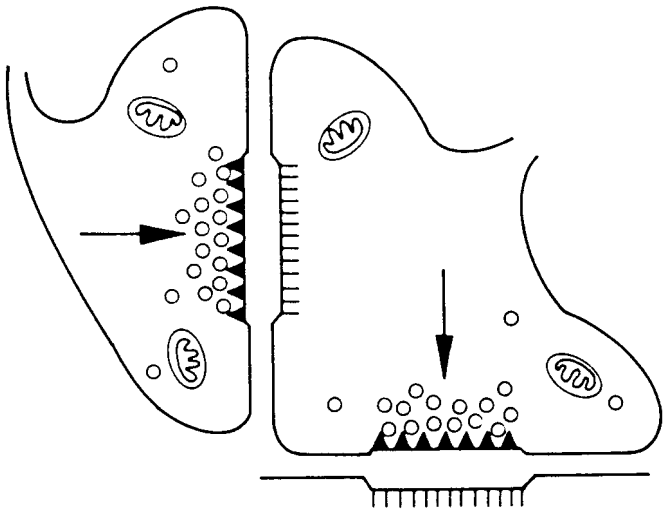


Figure D.1: Schematic drawing of a higher-order synapse. Upper part: terminals of the axons of two sending neurons, bottom: surface of a dendrite of the receiving neuron.

neurotransmitter, so that equation (1.4) will be replaced by:

$$I_k(t) = p_k(t) J_k S_k(t - \tau_k) + \sum_{l=1}^{N} \hat{p}_{kl}(t) \hat{J}_{kl} S_k(t - \tau_k) S_l(t - \tau_l)$$

in which the new variables have the following meaning: $\hat{p}_{kl}(t)$ is a random variable deciding whether two simultaneously arriving action potentials from neurons k and l are succesful in releasing neurotransmitter, $\hat{J}_{kl}$ defines the resulting current induced by the second order synapse, if transmitter release takes place. Assume $\hat{J}_{kk} = 0$ for all k.

- Which will be the new equation to replace equation (1.12), if we take into account the existence of the above type of higher order synapse ?
- Assume (without proof) that the simple scaling argument again applies, which allows us to replace all summations over different current contributions by their noise averages. Which equation do we find to replace (1.23) ?
- Perform the simplifications that lead to the graded response neurons. What will these equations be now ?
- The same question for the McCulloch-Pitts neurons.

2. Graded Response Neurons

Consider two identical coupled graded response neurons, without self-interactions, with $\tau = \rho = 1$:

$$\frac{d}{dt}U_1(t) = J_a g[U_2 - U^*] - U_1 \qquad \frac{d}{dt}U_2(t) = J_b g[U_1 - U^*] - U_2$$

- Assume $g[x] > 0\ \forall x$ (like for instance $g[x] = \frac{1}{2}[1 + \mathrm{erf}(x)]$). Show:
  (a) if $J_a = J_b$ there exists a solution of the form $U_1(t) = U_2(t)\ \forall t$.
  (b) if $J_a \neq J_b$ there does not exists a solution of the form $U_1(t) = U_2(t)\ \forall t$.

Now choose $g[x] = \theta[x]$ (no noise), $J_a = J_b = J \neq 0$. Simplify the equations by tranforming to new variables:

$$U_1 = Ju_1 \qquad U_2 = Ju_2 \qquad U^* = Ju^*$$

so that the new equations become

$$\frac{d}{dt}u_1(t) = \theta[u_2 - u^*] - u_1 \qquad \frac{d}{dt}u_2(t) = \theta[u_1 - u^*] - u_2$$

- Solve these equations for the four relevant regions of the (u_1, u_2) plane, (I) $u_1 > u^*,\ u_2 > u^*$ (II) $u_1 < u^*,\ u_2 > u^*$ (III) $u_1 < u^*,\ u_2 < u^*$ (IV) $u_1 > u^*,\ u_2 < u^*$, and draw the trajectories in the (u_1, u_2) plane, for $u^* < 1$.
- Do the same for $u^* > 1$.
- What can you conclude about the dependence of the system's behaviour on the height of the neural threshold ?

3. Coupled Oscillators

Consider three coupled oscillators, without self-interactions:

$$\frac{d}{dt}\phi_1(t) = \omega_1 + J_{12}\sin[\phi_2(t) - \phi_1(t)] + J_{13}\sin[\phi_3(t) - \phi_1(t)]$$

$$\frac{d}{dt}\phi_2(t) = \omega_2 + J_{21}\sin[\phi_1(t) - \phi_2(t)] + J_{23}\sin[\phi_3(t) - \phi_2(t)]$$

$$\frac{d}{dt}\phi_3(t) = \omega_3 + J_{31}\sin[\phi_1(t) - \phi_3(t)] + J_{32}\sin[\phi_2(t) - \phi_3(t)]$$

Explain the significance of the quantities which arise in these euqtions.

From now on assume that $\omega_1 = \omega_2 = \omega_3 = \omega$. Show that the above system of equations admits solutions such that

$$\phi_3(t) - \phi_2(t) = m\pi, \quad m = 0, \pm 1, \pm 2, \ldots$$

if and only if $J_{31} = (-1)^m J_{21}$. What do these solutions represent in terms of firing coherence. Assume that the above condition holds for some *even* integer m, and that $J_{ik} = J$ for all i, k. Show that

$$\frac{d}{dt}(\phi_1 + 2\phi_2) = 3\omega \qquad \frac{d}{dt}(\phi_1 - \phi_2) = -3J\sin(\phi_1 - \phi_2)$$

Finally, suppose that $J > 0$ and that *initially* $\phi_1 - \phi_2$ is not a multiple of π. Show, by considering the phase diagram pertaining to the above differential equation for $\phi_1 - \phi_2$, or otherwise, that as $t \to \infty$ all three oscillators will fire coherently, i.e. synchronously. What happens if $J < 0$?

4. Elementary Operations and Model Neurons

Consider the task $f : \{0,1\}^3 \to \{0,1\}$ given by

$$f(x_1, x_2, x_3) = \begin{cases} 1, & (x_1, x_2, x_3) = (1,0,1) \\ 1, & (x_1, x_2, x_3) = (1,1,1) \\ 0, & \text{otherwise} \end{cases}$$

- Show, by geometrical considerations, or otherwise, that f can be realised by a single McCulloch-Pitts neuron. Give suitable values for weights and threshold explicitly.

If we take into account higher order synapses, the operation of a McCulloch-Pitts neuron becomes somewhat more sophisticated:

$$S(t + \Delta) = \theta[\sum_k J_k x_k + \sum_{k,l} \hat{J}_{kl} x_k x_l - U]$$

- Can we perform the XOR operation $(x_1, x_2) \to \mathrm{XOR}(x_1, x_2)$ with $(x_1, x_2) \in \{0,1\}^2$, which a McCulloch-Pitts neuron cannot realise, with a single neuron like this ?

5. The Parity Operation

Define the parity operation

$$M : \{0,1\}^K \to \{0,1\} \qquad M(\boldsymbol{x}) = \frac{1}{2}\left[1 + (-1)^{\sum_{i=1}^K x_i}\right]$$

M indicates whether ($M = 1$) or not ($M = 0$) the number of $+1$ components of the input vector $\boldsymbol{x}$ is even.

- Show that for $K = 2$ we find $M(x_1, x_2) = \neg\mathrm{XOR}(x_1, x_2)$.
- Prove that for $K \geq 2$ the parity operation cannot be performed by a single McCulloch-Pitts neuron

$$S(\boldsymbol{x}) = \theta[\sum_{i=1}^K J_i x_i - U]$$

6. Symmetries

We know that a two-layer feed-forward network of McCulloch-Pitts neurons, with $L = |\Omega^+| \leq 2^K$ neurons in the 'hidden' layer can perform any transformation $M : \{0,1\}^K \to \{0,1\}$, provided all connections and thresholds are properly choosen. Here $|\Omega^+|$ denotes the number of vectors in the set Ω^+. However, in the case where there are symmetries in the operation M, the number of hidden neurons L can be considerably less. An example of such an operation is the Parity Operation (see above). Here the operation M is invariant under all permutations of the indices $\{1, \ldots, K\}$ of the input vector $\boldsymbol{x}$.

- We take $K = 2$. $\Omega^+ = \{(0,0),(1,1)\}$, so the upper bound for L is 2. Construct a two-layer feed-forward network of McCulloch-Pitts neurons with $L = 2$ that performs the parity operation, not as a realisation of a look-up table, but by using the fact that, due to the permutation symmetry, $M(\boldsymbol{x})$ can be written as a function only of $x_1 + x_2$. Hints: write $y_i(\boldsymbol{x}) = \theta[x_1 + x_2 - U_i]$, and use a graphical representation of what the output neuron $S(y_1, y_2)$ is required to do in order to find its separating plane.

- Choose $K = 3$. Since $\Omega^+ = \{(0,0,0),(0,1,1),(1,0,1),(1,1,0)\}$ the upper bound for L is 4. Construct a two-layer feed-forward network of McCulloch-Pitts neurons with $L = 3$ that performs the parity operation.

- Give the network construction for general $K \geq 2$ (without proof), by generalising the results obtained for $K = 2, 3$. Show that we need only K hidden neurons.

- A tough one: give the corresponding proof, for general K, that the construction performs the task M.

7. Learning the Unlearnable

We now study what happens when a perceptron is being trained on a task that is not linearly separable, like XOR(x_1, x_2). In ± 1 representation, and with the convention $x_0 = 1$ (the dummy input), the task T and the Ising perceptron σ are defined as

$$T : \{-1,1\}^3 \to \{-1,1\}, \qquad T(x_0, x_1, x_2) = -x_1 x_2$$

$$\sigma : \{-1,1\}^3 \to \{-1,1\}, \qquad \sigma(x_0, x_1, x_2) = \ \mathrm{sgn}[\boldsymbol{J} \cdot \boldsymbol{x}], \qquad \boldsymbol{J} = (J_0, J_1, J_2)$$

with $\Omega = \{\boldsymbol{x} \in \{-1,1\}^3 |\ x_0 = 1\}$ and $p(\boldsymbol{x}) = \frac{1}{4}\ \forall \boldsymbol{x} \in \Omega$ (note: there are four input vectors in Ω). In the limit of small learning rate $\epsilon \to 0$ the learning process is described by equation (2.13):

$$\frac{d}{dt}\boldsymbol{J} = \frac{1}{2}\langle \boldsymbol{x}\,[T(\boldsymbol{x}) - \ \mathrm{sgn}(\boldsymbol{J} \cdot \boldsymbol{x})]\rangle_\Omega$$

- Calculate $\langle \boldsymbol{x} T(\boldsymbol{x})\rangle_\Omega$.

- Prove that $|\boldsymbol{J}|$ decreases monotonically with time.

- Show that the dynamic equation (2.13) is a gradient descent on a surface $E(\boldsymbol{J})$. What is the meaning of $E(\boldsymbol{J})$?

- Prove that $\lim_{t\to\infty} |\boldsymbol{J}(t)| = 0$.

8. Application of the CLT to $\boldsymbol{W} \cdot \boldsymbol{x}$

A sufficient (although not necessary) condition for the inner product $\boldsymbol{W} \cdot \boldsymbol{x}$ to acquire a Gaussian probability distribution in the limit $N \to \infty$, with $\boldsymbol{x} \in \{-1, 1\}^N$ and $p(\boldsymbol{x}) = 2^{-N}$ $\forall \boldsymbol{x} \in \{-1, 1\}^N$ (i.e. for the the Central Limit Theorem to apply), is Lindeberg's Condition:

$$\forall \epsilon > 0 : \qquad \lim_{N \to \infty} \sum_{i=1}^{N} \theta \left[W_i^2 - \epsilon \sum_{k=1}^{N} W_k^2 \right] = 0$$

A necessary (although not sufficient) condition for the inner product $\boldsymbol{W} \cdot \boldsymbol{x}$ to acquire a Gaussian probability distribution in the limit $N \to \infty$ is

$$\lim_{N \to \infty} \frac{\sum_{i=1}^{N} W_i^4}{[\sum_{i=1}^{N} W_i^2]^2} = 0$$

Here we will look more closely at trivial and non-trivial teacher vectors $\boldsymbol{W} = (W_1, \ldots, W_N)$ for which the CLT does not apply.

- Show that both conditions are violated if $W_1 = 1$ and $W_i = 0$ $\forall i > 1$.
- Same question for $W_i = 1$ for $i \leq n$ and $W_i = 0$ $\forall i > n$, for every fixed $n \geq 1$ (fixed means: not dependent on N).
- Same question for $W_k = e^{-k}$.
- Same question for $W_k = 1/k$.
- Show that both conditions are sastisfied for $W_k = 1/\sqrt{k}$.

You may use:

$$\sum_{k=1}^{\infty} z^k = \frac{z}{1-z} \quad (|z| < 1), \qquad \sum_{k=1}^{\infty} \frac{1}{k^2} = \frac{1}{6}\pi^2, \qquad \sum_{k=1}^{\infty} \frac{1}{k^4} = \frac{11}{180}\pi^4, \qquad \sum_{k=1}^{\infty} \frac{1}{k} = \infty$$

9. Forgetful Perceptrons

Consider a perceptron σ with N ordinary input variables in ± 1 representation and with the convention of the dummy input $x_0 = 1$, learning a task T:

$$T : \{-1, 1\}^{N+1} \to \{-1, 1\}$$

$$\sigma : \{-1, 1\}^{N+1} \to \{-1, 1\}, \qquad \sigma(x_0, \ldots, x_N) = \text{ sgn}(\boldsymbol{J} \cdot \boldsymbol{x})$$

Assume $\Omega = \{\boldsymbol{x} \in \{-1, 1\}^{N+1} |\ x_0 = 1\}$ and $p(\boldsymbol{x}) = 2^{-N}$ $\forall \boldsymbol{x} \in \Omega$ (note: there are 2^N input vectors in Ω). In the original perceptron learning rule we add a decay term for the interactions, which tends to erase information previously accumulated:

$$\Delta \boldsymbol{J} = \frac{1}{2} \boldsymbol{x} \left[T(\boldsymbol{x}) - \text{ sgn}(\boldsymbol{J} \cdot \boldsymbol{x}) \right] - \eta \boldsymbol{J}$$

- Introduce a learning rate ϵ in the usual way: $\boldsymbol{J}(t + \epsilon) = \boldsymbol{J}(t) + \epsilon \Delta \boldsymbol{J}$. Derive the corresponding continuous-time equation to replace (2.13), by taking the limit $\epsilon \to 0$.

- Show that this equation describes gradient descent on a surface $E(\boldsymbol{J})$, and prove that $E(\boldsymbol{J})$ is bounded from below.
- Assume that T is linearly separable, i.e. $T(\boldsymbol{x}) = \text{sgn}(\boldsymbol{W}\cdot\boldsymbol{x})\ \forall \boldsymbol{x}\in\Omega$. Show that $\lim_{t\to\infty}\boldsymbol{W}\cdot\boldsymbol{J}(t)\geq 0$.

From now on consider only linearly separable tasks and perceptrons without thresholds, i.e. $W_0 = J_0 = 0$, and choose $|\boldsymbol{W}| = 1$.

- Define, following the analysis in section 2.5, the two quantities $J = |\boldsymbol{J}|$ and $\omega = \hat{\boldsymbol{J}}\cdot\boldsymbol{W}$, with $\boldsymbol{J} = J\hat{\boldsymbol{J}}$. Derive the differential equations that replace (2.44) and (2.45). Note that (2.45) is not affected by the decay term.
- Assume the two inner products $u = \boldsymbol{W}\cdot\boldsymbol{x}$ and $v = \hat{\boldsymbol{J}}\cdot\boldsymbol{x}$ to have a Gaussian joint probability distribution, and find the equations that replace (2.51) and (2.52).
- Show that (2.54) is replaced by $e^{\eta t}J(t)[1+\omega(t)] = J(0)[1+\omega(0)]$.
- Show that the evolution in time of the macroscopic observable ω, following $\omega(0) = 0$, is now given by

$$\frac{1}{\eta}\left[e^{\eta t}-1\right] = J(0)\sqrt{\frac{\pi}{2}}\left\{\log\left[\frac{1+\omega}{1-\omega}\right]^{\frac{1}{2}}+\frac{\omega}{1+\omega}\right\}$$

- Show how for $\eta \ll 1$ and times $t \ll \eta^{-\frac{1}{2}}$ we recover the old result (2.59). Hint: use the expansion $e^x = 1 + x + \frac{1}{2}x^2 + \mathcal{O}(x^3)$ for small x.
- Show that for any $\eta > 0$: $\lim_{t\to\infty}\omega(t) = 1$ and $\lim_{t\to\infty}J(t) = 0$. Which are the advantages and disadvantages of forgetting by weight decay ?

10. Examples for Noiseless Parallel Dynamics

Consider noiseless recurrent networks with parallel dynamics. Assume that the post-synaptic potentials $h_i = \sum_j J_{ij}\sigma_j + w_i$ are always non-zero.

- Consider example (ii) in the lecture notes: $J_{ij} = \frac{J}{N}$, $w_i = 0$. Write the Lyapunov function L (3.7) in terms of the average activity $m(\boldsymbol{\sigma}) = \frac{1}{N}\sum_i\sigma_i$, and verify that L decreases monotonically. Which values for L are obtained in the different attractors ?
- Same questions for example (iii) in the lecture notes: $J_{ij} = \frac{J}{N}$, $w_i = w \neq 0$.

Now turn to example (iv) in the lecture notes: $J_{ij} = \frac{J}{N}$, $w_i \in \{-w, w\}$. Define the two sub-networks $I_+ = \{i|\ w_i = w\}$ and $I_- = \{i|\ w_i = -w\}$, assume them to be equally large: $|I_+| = |I_-| = \frac{1}{2}N$. Define also the corresponding average activities

$$m_+(\boldsymbol{\sigma}) = \frac{2}{N}\sum_{i\in I_+}\sigma_i \qquad m_-(\boldsymbol{\sigma}) = \frac{2}{N}\sum_{i\in I_-}\sigma_i$$

- Calculate $m_+(t)$ and $m_-(t)$ for $t > 0$ (along the lines of the lecture notes)
- Write the Lyapunov function L (3.7) in terms of the two average activities $m_\pm(\boldsymbol{\sigma})$ in the sub-networks, and verify that L decreases monotonically. Which values for L are obtained in the different attractors ?

11. Examples for Noiseless Sequential Dynamics

Consider noiseless recurrent networks with sequential dynamics. Choose $J_{ij} = \frac{J}{N}$, $w_i = 0$ and N odd. The average activity in the system is defined as usual: $m(t) = \frac{1}{N}\sum_i \sigma_i(t)$. Assume $m(0) \neq 0$.

- Show that for $J > 0$: $m(\infty) = \lim_{t\to\infty} m(t) = \ \mathrm{sgn}[m(0)]$ (as for parallel dynamics)
- Show that for $J < 0$, on the other hand, the behaviour is completely different from the corresponding system with parallel dynamics.
- Calculate $\lim_{N\to\infty} m(\infty)$ for $J < 0$.

12. Lyapunov Functions

Consider networks with *anti*-symmetric synaptic interactions, i.e. $J_{ij} = -J_{ji}$ $(\forall ij)$, with $w_i = 0$ $(\forall i)$ and parallel deterministic dynamics.

- Show that $L(\boldsymbol{\sigma})$ (3.7) is also a Lyapunov function for networks with anti-symmetric interactions.
- Prove that anti-symmetric networks will always evolve into a period-4 limit-cycle.

13. Information Storage Through the Creation of Attractors

Consider noiseless recurrent networks with parallel dynamics: $\sigma_i(t{+}1) = \ \mathrm{sgn}[\sum_j J_{ij}\sigma_j(t)]$ $(\forall i)$. The p vectors $\boldsymbol{\xi}^\mu = (\xi_1^\mu, \ldots, \xi_N^\mu) \in \{-1, 1\}^N$ $(\mu = 1, \ldots, p)$ represent patterns (information) to be stored and retrieved. Assume these patterns to be mutually orthogonal and that p to be finite. We define the p pattern overlaps $m_\mu(\boldsymbol{\sigma}) = \frac{1}{N}\sum_i \xi_i^\mu \sigma_i \in [-1, 1]$. Choose the synaptic interactions

$$J_{ij} = \frac{1}{N}\sum_{\mu=1}^{p} \xi_i^\mu \xi_j^\mu$$

- Give a condition on the initial overlaps $\{m_\mu(0)\}$ sufficient to guarantee that $\boldsymbol{\sigma}(1) = \boldsymbol{\xi}^\lambda$.
- Show that for $N \to \infty$ all pattern vectors $\boldsymbol{\xi}^\mu$ are fixed-point attractors, by demonstrating that the above condition is fulfilled for a set of states close to these patterns.

Now choose

$$J_{ij} = \frac{1}{N}\sum_{\mu=1}^{p-1} \xi_i^{\mu+1}\xi_j^\mu + \frac{1}{N}\xi_i^1\xi_j^p$$

- Give a condition on the initial overlaps $\{m_\mu(0)\}$ sufficient to guarantee that $\boldsymbol{\sigma}(1) = \boldsymbol{\xi}^\lambda$.
- Show that for $N \to \infty$ there exists a stable period-p limit cycle attractor of the form $\boldsymbol{\xi}^1 \to \boldsymbol{\xi}^2 \to \cdots \to \boldsymbol{\xi}^{p-1} \to \boldsymbol{\xi}^p \to \boldsymbol{\xi}^1 \to \cdots$

14. Detailed Balance

Consider the general Markov chain (3.20).

- Show that from $p_t(\boldsymbol{\sigma})$ being a probability distribution it follows that the transition matrix must have the property $\sum_{\boldsymbol{\sigma}} W[\boldsymbol{\sigma};\boldsymbol{\sigma}'] = 1$ for all $\boldsymbol{\sigma}'$.

- Show that a distribution $p(\boldsymbol{\sigma})$ satisfying the so-called detailed-balance condition

$$W[\boldsymbol{\sigma};\boldsymbol{\sigma}']p(\boldsymbol{\sigma}') = W[\boldsymbol{\sigma}';\boldsymbol{\sigma}]p(\boldsymbol{\sigma}) \qquad (\forall \boldsymbol{\sigma},\boldsymbol{\sigma}')$$

  gives a stationary state of the Markov chain.

- Show that for the stochastic parallel dynamics networks (3.21) with symmetric interactions the stationary state is given by

$$p_\infty(\boldsymbol{\sigma}) = C.e^{\beta\sum_i \sigma_i w_i}\prod_i \cosh\beta[\sum_j J_{ij}\sigma_j + w_i]$$

  (where C is a constant), by showing that it satisfies the detailed balance condition.

- Show that for the stochastic sequential dynamics networks (3.23) with symmetric interactions and $J_{ii} = 0$ $(\forall i)$ the stationary state is given by

$$p_\infty(\boldsymbol{\sigma}) = C.e^{\beta\left[\frac{1}{2}\sum_{ij}\sigma_i J_{ij}\sigma_j + \sum_i \sigma_i w_i\right]}$$

  (where C is a constant), by showing that it satisfies the detailed balance condition.

15. Analysis of Attractor Models by Mean Field Approximation

Consider the attractor models

$$h_i(\boldsymbol{\sigma}) = \sum_j J_{ij}\sigma_j \qquad J_{ij} = \frac{1}{N}\sum_{\mu\nu}\xi_i^\mu A_{\mu\nu}\xi_j^\nu$$

Define pattern overlaps $m_\mu(\boldsymbol{\sigma}) = \frac{1}{N}\sum_i \xi_i^\mu\sigma_i$, and averages $\langle f(\boldsymbol{\sigma})\rangle_t = \sum_{\boldsymbol{\sigma}} p_t(\boldsymbol{\sigma})f(\boldsymbol{\sigma})$ $(\boldsymbol{\sigma}\in\{-1,1\}^N)$. The mean field approximation consists of replacing all local fields by their averages, giving $h_i(\boldsymbol{\sigma}) \to \sum_{\mu\nu}\xi_i^\mu A_{\mu\nu}\langle m_\nu(\boldsymbol{\sigma})\rangle_t$. Define the continuous-time sequential dynamics:

$$\frac{d}{dt}p_t(\boldsymbol{\sigma}) = \sum_i\{w_i(F_i\boldsymbol{\sigma})p_t(F_i\boldsymbol{\sigma}) - w_i(\boldsymbol{\sigma})p_t(\boldsymbol{\sigma})\} \qquad w_i(\boldsymbol{\sigma}) = \frac{1}{2}[1-\sigma_i\tanh[\beta h_i(\boldsymbol{\sigma})]] \quad \text{(D.1)}$$

(with the neuron flip-operation F_i).

- Derive for the process (D.1) the general relation

$$\frac{d}{dt}\langle f(\boldsymbol{\sigma})\rangle_t = \langle\sum_i w_i(\boldsymbol{\sigma})\,[f(F_i\boldsymbol{\sigma}) - f(\boldsymbol{\sigma})]\rangle_t$$

- Apply the above result to the overlaps $m_\mu(\boldsymbol{\sigma})$ and use the mean-field approximation to derive the macroscopic laws

$$\frac{d}{dt}m_\mu(t) = \frac{1}{N}\sum_k \xi_k^\mu \tanh[\beta \sum_{\lambda\nu} \xi_k^\lambda A_{\lambda\nu} m_\nu(t)] - m_\mu(t) \qquad \text{with} \quad m_\mu(t) = \langle m_\mu(\boldsymbol{\sigma})\rangle_t$$

Now turn to discrete-time parallel dynamics:

$$p_{t+1}(\boldsymbol{\sigma}) = \sum_{\boldsymbol{\sigma}'} W[\boldsymbol{\sigma};\boldsymbol{\sigma}']p_t(\boldsymbol{\sigma}') \qquad W[\boldsymbol{\sigma};\boldsymbol{\sigma}'] = \prod_i \frac{e^{\beta\sigma_i h_i(\boldsymbol{\sigma}')}}{2\cosh[\beta h_i(\boldsymbol{\sigma}')]} \tag{D.2}$$

- Derive for the process (D.2) the general relation:

$$\langle f(\boldsymbol{\sigma})\rangle_{t+1} = \sum_{\boldsymbol{\sigma}'} \frac{p_t(\boldsymbol{\sigma}')}{\prod_i 2\cosh[\beta h_i(\boldsymbol{\sigma}')]} \sum_{\boldsymbol{\sigma}} f(\boldsymbol{\sigma}) \prod_i e^{\beta\sigma_i h_i(\boldsymbol{\sigma}')}$$

- Prove the identity (for $\{A_i\}$ not dependent on $\boldsymbol{\sigma}$):

$$\sum_{\boldsymbol{\sigma}} \sigma_k \prod_i e^{\sigma_i A_i} = \tanh(A_k) \prod_i [2\cosh(A_i)]$$

- Apply the above results to the overlaps $m_\mu(\boldsymbol{\sigma})$ and use the mean-field approximation to derive the macroscopic laws

$$m_\mu(t{+}1) = \frac{1}{N}\sum_k \xi_k^\mu \tanh[\beta \sum_{\lambda\nu} \xi_k^\lambda A_{\lambda\nu} m_\nu(t)] \qquad \text{with} \quad m_\mu(t) = \langle m_\mu(\boldsymbol{\sigma})\rangle_t$$

16. Attractor Models with Unequal Embedding Strengths

We apply the results of the previous exercise to the special case where patterns are stored according to a Hebbian rule, but with different (positive) embedding strengths w_μ: $A_{\mu\nu} = w_\mu\delta_{\mu\nu}$, or

$$J_{ij} = \frac{1}{N}\sum_\mu w_\mu \xi_i^\mu \xi_j^\mu$$

Assume random patterns and consider large networks ($N \to \infty$).

- Show that the macroscopic laws have (pure) solutions of the form $m_\mu(t) = m(t)\delta_{\mu\rho}$, what do they represent ? Show that the stationary value $m = \lim_{t\to\infty} m(t)$ is a solution of the transcendental equation $m = \tanh(\beta w_\rho m)$.

- Show that the previous result implies that there exists a critical value for the noise level $T = \beta^{-1}$ above which only the trivial pure stationary state $m = 0$ is possible, what is this value ?