

1 Introduction: Neural Information Processing

Our brains perform sophisticated information processing tasks, using hardware and operation rules which are quite different from the ones on which conventional computers are based. The processors in the brain, the neurons (see figure 1), are rather noisy elements¹ which operate in parallel. They are organised in dense networks, the structure of which can vary from very regular to almost amorphous (see figure 2), and they communicate signals through a huge number of inter-neuron connections (the so-called synapses). These connections represent the ‘program’ of a network. By continuously updating the strengths of the connections, a network as a whole can modify and optimise its ‘program’, ‘learn’ from experience and adapt to changing circumstances.

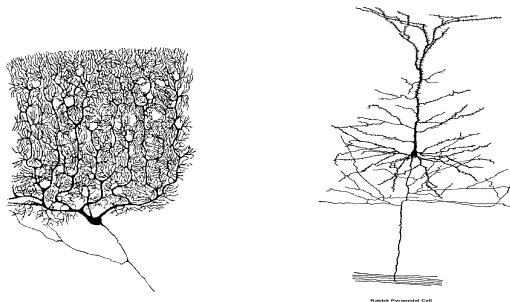


Figure 1: Left: a Purkinje neuron in the human cerebellum. Right: a pyramidal neuron of the rabbit cortex. The black blobs are the neurons, the trees of wires fanning out constitute the input channels (or dendrites) through which signals are received which are sent off by other firing neurons. The lines at the bottom, bifurcating only modestly, are the output channels (or axons).

From an engineering point of view neurons are in fact rather poor processors, they are slow and unreliable (see the table below). In the brain this is overcome by ensuring that always a very large number of neurons are involved in any task, and by having them operate in parallel, with many connections. This is in sharp contrast to conventional computers, where operations are as a rule performed sequentially, so that failure of any part of the chain of operations is usually fatal. Furthermore, conventional computers execute a detailed specification of orders, requiring the programmer to know exactly which data can be expected and how to respond. Subsequent changes in the actual situation, not foreseen by the programmer, lead to trouble. Neural networks, on the other hand, can adapt to changing circumstances. Finally, 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.

¹By this we mean that their output signals are to some degree subject to random variation; they exhibit so-called spontaneous activity which appears not to be related to the information processing task they are involved in.

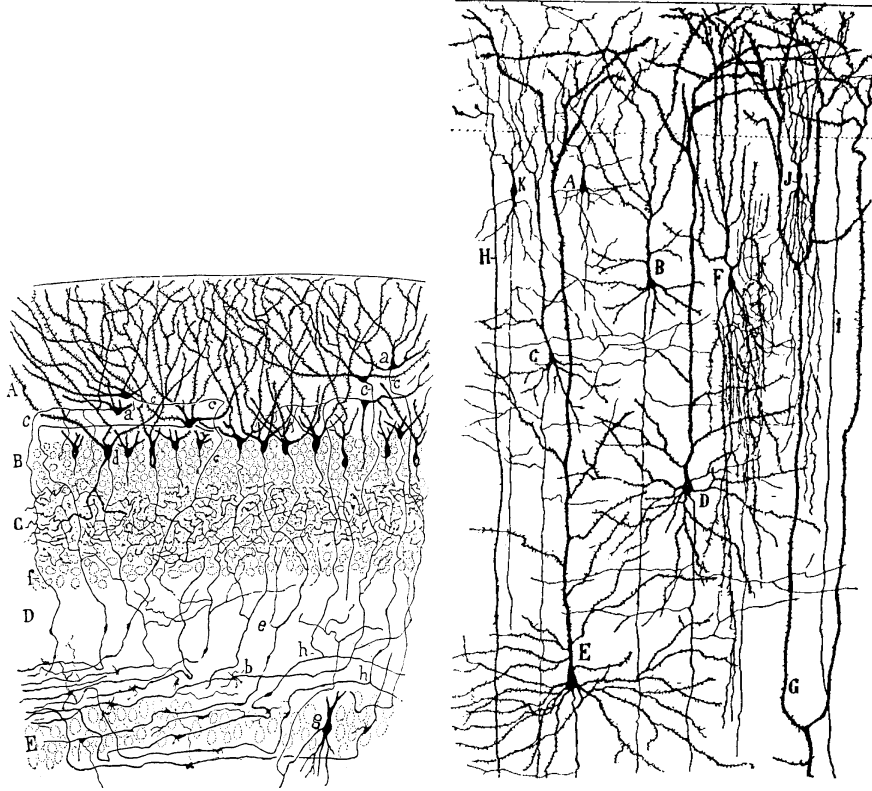


Figure 2: Left: a section of the human cerebellum. Right: a section of the human cortex. Note that the staining method used to produce such pictures colours only a reasonably modest fraction of the neurons present, so in reality these networks are far more dense.

Roughly speaking, conventional computers can be seen as the appropriate tools for performing well-defined and rule-based information processing tasks, in stable and safe environments, where all possible situations, as well as how to respond in every situation, are known beforehand. Typical tasks fitting these criteria are e.g. brute-force chess playing, word processing, keeping accounts and rule-based (civil servant) decision making. Neural information processing systems, on the other hand, are superior to conventional computers in dealing with real-world tasks, such as e.g. communication (vision, speech recognition), movement coordination (robotics) and experience-based decision making (classification, prediction, system control), where data are often messy, uncertain or even inconsistent, where the number of possible situations is infinite and where perfect solutions are for all practical purposes non-existent.

One can distinguish three types of motivation for studying neural networks. Biologists, physiologists, psychologists and to some degree also philosophers aim at understanding information processing in real biological nervous tissue. They study models, mathematically and through computer simulations, which are preferably close to what is being observed experimentally, and try to understand the global properties and functioning of brain regions.

conventional computers	biological neural networks
processors <i>operation speed</i> $\sim 10^8 \text{ Hz}$ <i>signal/noise</i> $\sim \infty$ <i>signal velocity</i> $\sim 10^8 \text{ m/sec}$ <i>connections</i> ~ 10	neurons <i>operation speed</i> $\sim 10^2 \text{ Hz}$ <i>signal/noise</i> ~ 1 <i>signal velocity</i> $\sim 1 \text{ m/sec}$ <i>connections</i> $\sim 10^4$
sequential operation program & data external programming	parallel operation connections, neuron thresholds self-programming & adaptation
hardware failure: fatal no unforeseen data	robust against hardware failure messy, unforeseen data

Engineers and computer scientists would like to understand the principles behind neural information processing in order to use these for designing adaptive software and artificial information processing systems which can also ‘learn’. They use highly simplified neuron models, which are again arranged in networks. As their biological counterparts, these artificial systems are not programmed, their inter-neuron connections are not prescribed, but they are ‘trained’. They gradually ‘learn’ to perform tasks by being presented with examples of what they are supposed to do. The key question then is to understand the relationships between the network performance for a given type of task, the choice of ‘learning rule’ (the recipe for the modification of the connections) and the network architecture. Secondly, engineers and computer scientists exploit the emerging insight into the way real (biological) neural networks manage to process information efficiently in parallel, by building artificial neural networks in hardware, which also operate in parallel. These systems, in principle, have the potential of being incredibly fast information processing machines.

Finally, it will be clear that, due to their complex structure, the large numbers of elements involved, and their dynamic nature, neural network models exhibit a highly non-trivial and rich behaviour. This is why also theoretical physicists and mathematicians have become involved, challenged as they are by the many fundamental new mathematical problems posed by neural network models. Studying neural networks as a mathematician is rewarding in two ways. The first reward is to find nice applications for one’s tools in biology and engineering. It is fairly easy to come up with ideas about how certain information processing tasks could be performed by (either natural or synthetic) neural networks; by working out the mathematics, however, one can actually

quantify the potential and restrictions of such ideas. Mathematical analysis further allows for a systematic design of new networks, and the discovery of new mechanisms. The second reward is to discover that one's tools, when applied to neural network models, create quite novel and funny mathematical puzzles. The reason for this is the 'messy' nature of these systems. Neurons are not at all well-behaved: they are microscopic elements which do not live on a regular lattice, they are noisy, they change their mutual interactions all the time, etc.

Since this paper aims at no more than sketching a biased impression of a research field, I will not give references to research papers along the way, but mention textbooks and review papers in the final section, for those interested.

2 From Biology to Mathematical Models

We cannot expect to solve mathematical models of neural networks in which all electro-chemical details are taken into account (even if we knew all such details perfectly). Instead we start by playing with simple networks of model neurons, and try to understand their basic properties first (i.e. we study elementary electronic circuitry before we volunteer to repair the video recorder).

2.1 From Biological Neurons to Model Neurons

Neurons operate more or less in the following way. The 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. A sufficiently strong local electric excitation, however, 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 around $+40$ 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 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 electric 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.

typical time-scales		typical sizes	
action potential:	$\sim 1msec$	cell body:	$\sim 50\mu m$
reset time:	$\sim 3msec$	axon diameter:	$\sim 1\mu m$
synapses:	$\sim 1msec$	synapse size:	$\sim 1\mu m$
pulse transport:	$\sim 5m/sec$	synaptic cleft:	$\sim 0.05\mu m$

The junction between an output channel (axon) of one neuron and an input channel (dendrite) of another neuron, is called synapse (see figure 3). The arrival at a synapse 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 selectively open 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^- channels the result is a

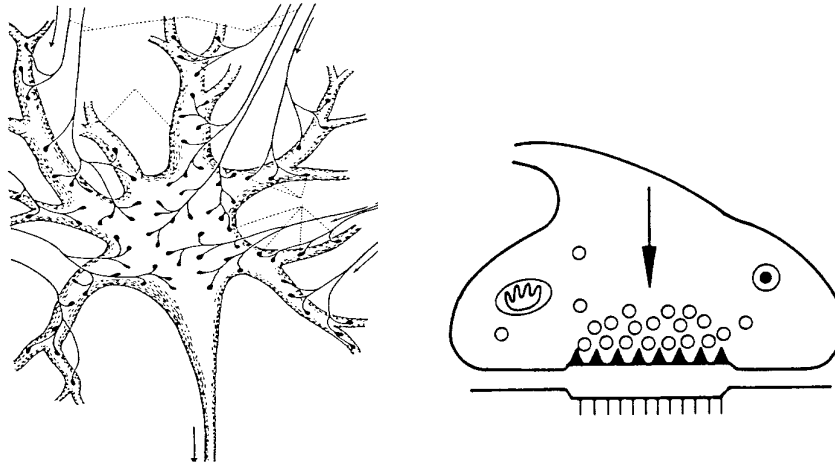


Figure 3: Left: drawing of a neuron. The black blobs attached to the cell body and the dendrites (input channels) represent the synapses (adjustable terminals which determine the effect communicating neurons will have on one another's membrane potential and firing state). Right: close-up of a typical synapse.

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. 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.

Whether or not the receiving neuron will actually be triggered into firing itself, will depend on the cumulative effect of all excitatory and inhibitory signals arriving, a detailed analysis of which requires also taking into account the electrical details of the dendrites. 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 post-synaptic 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, however, how this is realised at a chemical/electrical level. Most likely the amount of neurotransmitter in a synapse, available for release, and the effective

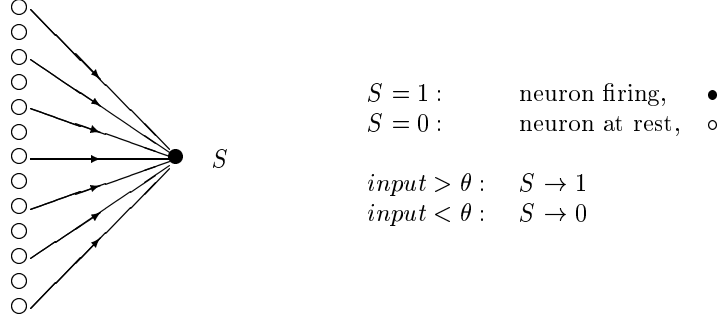


Figure 4: The simplest model neuron: a neuron’s firing state is represented by a single instantaneous binary state variable S , whose value is solely determined by whether or not its input exceeds a firing threshold.

contact surface of a synapse are modified.

The simplest caricature of a neuron is one where its possible firing states are reduced to just a single binary variable S , indicating whether it fires ($S = 1$) or is at rest ($S = 0$). See figure 4. Which of the two states the neuron will be in, is dictated by whether or not the total input it receives (i.e. the post-synaptic potential) does ($S \rightarrow 1$) or does not ($S \rightarrow 0$) exceed the neuron’s firing threshold, denoted by θ (if we forget about the noise). As a bonus this allows us to illustrate the collective firing state of networks by colouring the constituent neurons: firing = ●, rest = ○. We further assume the individual input signals to add up linearly, weighted by the strengths of the associated synapses. The latter are represented by real variables w_ℓ , whose sign denotes the type of interaction ($w_\ell > 0$: excitation, $w_\ell < 0$: inhibition) and whose absolute value $|w_\ell|$ denotes the magnitude of the interaction:

$$input = w_1 S_1 + \dots + w_N S_N$$

Here the various neurons present are labelled by subscripts $\ell = 1, \dots, N$. This rule indeed appears to capture the characteristics of neural communication. Imagine, for instance, the effect on the input of a quiescent neuron ℓ suddenly starting to fire:

$$S_\ell \rightarrow 1 : \quad input \rightarrow input + w_\ell \quad \begin{cases} w_\ell > 0 : & input \uparrow, \text{ excitation} \\ w_\ell < 0 : & input \downarrow, \text{ inhibition} \end{cases}$$

We now adapt these rules for each of our neurons. We indicate explicitly at which time t (for simplicity to be measured in units of one) the various neuron states are observed, we denote the synaptic strength at a junction $j \rightarrow i$ (where

j denotes the ‘sender’ and i the ‘receiver’) by w_{ij} , and the threshold of a neuron i by θ_i . This brings us to the following set of microscopic operation rules:

$$\begin{aligned} w_{i1}S_1(t) + \dots + w_{iN}S_N(t) &> \theta_i : S_i(t+1) = 1 \\ w_{i1}S_1(t) + \dots + w_{iN}S_N(t) &< \theta_i : S_i(t+1) = 0 \end{aligned} \quad (1)$$

These rules could either be applied to all neurons *at the same time*, giving so-called parallel dynamics, or to one neuron at a time (drawn randomly or according to a fixed order), giving so-called sequential dynamics.² Upon specifying the values of the synapses $\{w_{ij}\}$ and the thresholds $\{\theta_i\}$, as well as the initial network state $\{S_i(0)\}$, the system will evolve in time in a deterministic manner, and the operation of our network can be characterised by giving the states $\{S_i(t)\}$ of the N neurons at subsequent times, e.g.

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9
$t = 0 :$	1	1	0	1	0	0	1	0	0
$t = 1 :$	1	0	0	1	0	1	1	1	1
$t = 2 :$	0	0	1	1	1	0	0	0	1
$t = 3 :$	1	0	0	1	1	1	1	0	1
$t = 4 :$	0	1	1	1	0	0	1	0	1

or, equivalently, by drawing the neuron states at different times as a collection of coloured circles, according to the convention ‘firing’ = ●, ‘rest’ = ○, e.g.

$t = 0$	$t = 1$	$t = 2$	$t = 3$	$t = 4$

We have thus achieved a reduction of the operation of neural networks to a well-defined manipulation of a set of (binary) numbers, whose rules (1) can be seen as an extremely simplified version of biological reality. The binary numbers represent the states of the information processors (the neurons), and therefore describe the system *operation*. The details of the operation to be performed depend on a set of control parameters (synapses and thresholds), which must accordingly be interpreted as representing the *program*. Moreover, manipulating numbers brings us into the realm of mathematics; the formulation (1) describes a non-linear discrete-time dynamical system.

2.2 Universality of Model Neurons

Although it is not a priori clear that our equations (1) are not an oversimplification of biological reality, there are at least two reasons for not making things more complicated yet. First of all, solving (1) for arbitrary control parameters and nontrivial system sizes is already impossible, in spite of its apparent simplicity. Secondly, networks of the type (1) are found to be *universal* information

²Strictly speaking, we also need to specify a rule for determining $S_i(t+1)$ for the marginal case, where $w_{i1}S_1(t) + \dots + w_{iN}S_N(t) = \theta_i$. Two common ways of dealing with this situation are to either draw $S_i(t+1)$ at random from $\{0, 1\}$, or to simply leave $S_i(t+1) = S_i(t)$.

processing systems, in that (roughly speaking) they can perform any computation that can be performed by conventional digital computers, provided one chooses the synapses and thresholds appropriately.

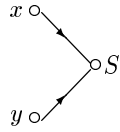
The simplest way to show this is by demonstrating that the basic logical units of digital computers, the operations AND: $(x, y) \rightarrow x \wedge y$, OR: $(x, y) \rightarrow x \vee y$ and NOT: $x \rightarrow \neg x$ (with $x, y \in \{0, 1\}$), can be built with our model neurons. Each logical unit (or ‘gate’) is defined by a so-called truth table, specifying its output for each possible input. All we need to do is to define for each of the above gates a model neuron of the type

$$\begin{aligned} w_1 x + w_2 y - \theta > 0 : S &= 1 \\ w_1 x + w_2 y - \theta < 0 : S &= 0 \end{aligned}$$

by choosing appropriate values of the control parameters $\{w_1, w_2, \theta\}$, which has the same truth table. This turns out to be fairly easy:

AND:

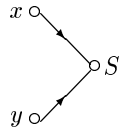
x	y	$x \wedge y$	$x + y - \frac{3}{2}$	S
0	0	0	$-3/2$	0
0	1	0	$-1/2$	0
1	0	0	$-1/2$	0
1	1	1	$1/2$	1



$w_1 = w_2 = 1$
 $\theta = \frac{3}{2}$

OR:

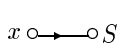
x	y	$x \vee y$	$x + y - \frac{1}{2}$	S
0	0	0	$-1/2$	0
0	1	1	$1/2$	1
1	0	1	$1/2$	1
1	1	1	$3/2$	1



$w_1 = w_2 = 1$
 $\theta = \frac{1}{2}$

NOT:

x	$\neg x$	$-x + \frac{1}{2}$	S
0	1	$1/2$	1
1	0	$-1/2$	0



$w_1 = -1$
 $\theta = -\frac{1}{2}$

This shows that we need not worry about a-priori restrictions on the types of tasks our simplified model networks (1) can handle.

Furthermore, one can also make statements on the architecture required. Provided we employ model neurons with potentially large numbers of input channels, it turns out that every operation involving binary numbers can in fact be performed with a feed-forward network of at most two layers. Again this is proven by construction. Every binary operation $\{0, 1\}^N \rightarrow \{0, 1\}^K$ can be reduced (split-up) into specific sub-operations M , each performing a separation of the input signals \mathbf{x} (given by N binary numbers) into two classes:

$$M : \{0, 1\}^N \rightarrow \{0, 1\}$$

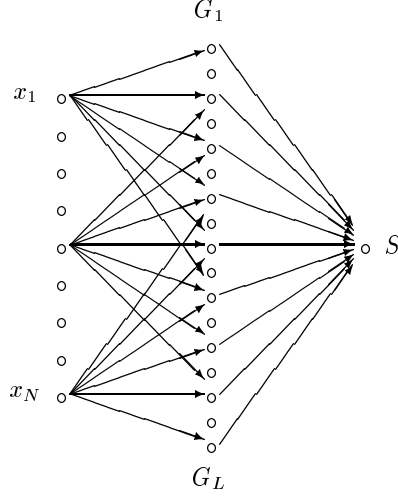


Figure 5: Universal architecture, capable of performing any classification $M : \{0, 1\}^N \rightarrow \{0, 1\}$, provided synapses and thresholds are chosen adequately.

(described by a truth table with 2^N rows). Each such M can be built as a neural realisation of a look-up exercise, where the aim is simply to check whether an $\mathbf{x} \in \{0, 1\}^N$ is in the set for which $M(\mathbf{x}) = 1$. This set is denoted by Ω , with $L \leq 2^N$ elements which we label as follows: $\Omega = \{\mathbf{y}_1, \dots, \mathbf{y}_L\}$. The basic tools of our construction are the so-called ‘grandmother-neurons’³ G_ℓ , whose sole task is to be on the look-out for one of the input signals $\mathbf{y}_\ell \in \Omega$:

$$\begin{aligned} w_1 x_1 + \dots + w_N x_N &> \theta : & G_\ell &= 1 \\ w_1 x_1 + \dots + w_N x_N &< \theta : & G_\ell &= 0 \end{aligned}$$

with $w_\ell = 2(2y_\ell - 1)$ and $\theta = 2(y_1 + \dots + y_N) - 1$. Inspection shows that with these definitions the output G_ℓ , upon presentation of input \mathbf{x} , is indeed (as required) given by

$$\begin{aligned} \mathbf{x} = \mathbf{y}_\ell : & \quad G_\ell = 1 \\ \mathbf{x} \neq \mathbf{y}_\ell : & \quad G_\ell = 0 \end{aligned}$$

Finally the outputs of the grandmother neurons are fed into a model neuron S , which is to determine whether or not one of the grandmother neurons is active:

$$\begin{aligned} y_1 + \dots + y_L &> 1/2 : & S &= 1 \\ y_1 + \dots + y_L &< 1/2 : & S &= 0 \end{aligned}$$

³This name was coined to denote neurons which only become active upon presentation of some unique and specific sensory pattern (visual or otherwise), e.g. an image of one’s grandmother. Such neurons were at some stage claimed to have been observed experimentally.

The resulting feed-forward network is shown in figure 5. For any input \mathbf{x} , the number of active neurons G_ℓ in the first layer is either 0 (leading to the final output $S = 0$) or 1 (leading to the final output $S = 1$). In the first case the input vector \mathbf{x} is apparently not in the set Ω , in the second case it apparently is. This shows that the network thus constructed performs the separation M .

2.3 Directions and Strategies

Here the field effectively splits in two. One route leading away from equation (1) aims at solving it with respect to the evolution of the neuron states, for increasingly complicated but prescribed choices of synapses and thresholds. Here the key phenomenon is *operation*, the central dynamical variables are the neurons, whereas synapses and thresholds play the role of parameters. The alternative route is to concentrate on the complementary problem: which are the possible modes of operation equation (1) would allow for, if we were to vary synapses and thresholds in a given architecture, and how can one find learning rules (rules for the modification of synapses and thresholds) that will generate values such that the resulting network will meet some specified performance criterion. Here the key phenomenon is *learning*, the central dynamical variables are the synapses and thresholds, whereas neuron states (or, more often, their statistics) induce constraints and operation targets.

Operation		Learning	
variables:	neurons	variables:	synapses, thresholds
parameters:	synapses, thresholds	parameters:	required neuron states

Although quite prominent, in reality this separation is, of course, not perfect; in the field of learning theory one often specifies neuron states only in part of the system, and solves for the remaining neuron states, and there even exist non-trivial but solvable models in which both neurons and synapses/thresholds evolve in time. In the following sections I will describe examples from both main problem classes.

A general rule in dealing with mathematical models, whether they describe phenomena in biology, physics, economics or any other discipline, is that one usually finds that the equations involved are most easily solved in extreme limits for the control parameters. This is also true for neural network models, in particular with respect to the system size N and the spatial distance over which the neurons are allowed to interact. Analysing models with just two or three neurons (on one end of the scale of sizes) is not much of a problem, but realistic systems happen to scale differently, both in biology (where even small brain regions are at least of size $N \sim 10^6$) and in engineering (where at least $N \sim 10^3$). Therefore one usually considers the opposite limit $N \rightarrow \infty$. In turn, one can only solve the equations describing infinitely large systems when either interactions are restricted to occur only between neighbouring neurons

(which is quite unrealistic), or when a large number (if not all) of the neurons are allowed to interact (which is a better approximation of reality).

The strategy of the model solver is then to identify global observables which characterise the system state at a macroscopic level (this is often the most difficult bit), and to calculate their values. For instance, in statistical mechanics one is not interested in knowing the positions and velocities of individual molecules in a gas, but rather in knowing the values of global observables like pressure; in modelling (and predicting) exchange rates we do not care about which individuals buy certain amounts of a currency, but rather in the sum over all such buyers. Which macroscopic observables constitute the natural language for describing the operation of neural networks turns out to depend strongly on their function or task (as might have been expected). If the exercise is carried out properly, and if the model at hand is sufficiently friendly, one will observe that in the $N \rightarrow \infty$ limit clean and transparent analytical relations emerge. This happens for various reasons. If there is an element of randomness involved (noise) it is clear that in finite systems we can only speak about the probability of certain averages occurring, whereas in the $N \rightarrow \infty$ limit one would find averages being replaced by exact expressions. Secondly, as soon as spatial structure of a network is involved, the limit $N \rightarrow \infty$ allows us to take continuum limits and to replace discrete systems by continuous ones.

The operation a neural network performs depends on its program: the choice made for architecture, synaptic interactions and thresholds (equivalently, on the learning rule used to generate these parameters). I will now give several examples involving different types of information processing tasks and, consequently, different types of analysis (although all share the reductionist strategy of calculating global properties from underlying microscopic laws).