AN INTRODUCTION TO NEURAL NETWORKS

David Barber
Aston University, Birmingham B4 7ET, U.K.

Tom Heskes
SNN, University of Nijmegen, The Netherlands

Keywords: neural networks, emergent behavior, statistical physics, learning, statistical inference, frequentist approaches, Bayesian methodology.

Summary:

This article gives a biased overview of neural-network research. Neural networks are computation models, inspired by information processing in the brain. Two aspects of neural networks are emphasized: the ability to generate complex behavior from many simple processing elements and the property to learn from data.
Emerging complex behavior is most prominent in classical neural networks like the Hopfield network and the Boltzmann machine. Techniques from statistical physics have been applied to understand their operation.
The learning capability of neural networks forms the basis of most practical applications of neural networks. In this context, neural networks have shifted from heuristic black boxes to advanced statistical tools. Learning can be interpreted as a statistical inference problem, which can be solved either using frequentist or Bayesian approaches.
Neural networks have been looked at from many different viewpoints. The cross-fertilization between neural networks and many other fields will continue to grow and improve neural networks, both as advanced statistical tools for solving practical problems and as computational models for understanding how the brain works.

Contents

1 What are neural networks?

The field of neural networks covers a large area ranging from theoretical neurobiology to statistical physics and machine learning. Covering in depth such a wide range of topics would be beyond the scope of this article, in which we prefer to give instead some insights into issues related to the history of the field, theoretical concepts that underpin the field, and then give an overview of more practical applications. There are many good textbooks available, covering much of the material in this review, see the references for details.
What is it that ties such seemingly disparate areas of research together? In all these areas there is a common interest in the properties of a (possibly very large) number of relatively simple processing units (neurons/spin-particles/elementary computing units) when they are coupled together. It is the belief that the emergent phenomenon when such simple units are coupled together is capable of explaining such complex effects as intelligence, memory, magnetism and can provide a useful basis for machine learning and computation. In such an interdisciplinary field, neural networks can take on somewhat different meanings, and the demands or emphasis of the research are correspondingly different. Following Churchland and Sejnowski, we can loosely categorise the demand/complaint pairs of different researchers, based on this underlying neural-network research as[1]

The Neuroscientist. Show me results of neuromodeling that help explain or predict experimental results.
Non neuroscientists do not know anything much about neuroscience even though they are doing "neural modeling".

The Psychologist. Show me results of neuromodeling that help explain or predict psychological functions and behaviour.
Non psychologists do not know anything much about the results from psychophysics and psychology even though they are modeling psychological capacities and performance.

The Computer Scientist. Show me results of neuromodeling that help understand the nature of computation and representation or that yield new ideas about these things.
Non computer scientists do not know anything much about electrical circuits, mathematical analyses, or existing theories of computation.

The Philosopher. Show me results of neuromodeling that are relevant to philosophical problems concerning the nature of knowledge, the self and the mind.
Non philosophers do not understand some of the useful, time saving, and agony-saving contributions of philosophers in constraining questions about how the mind works.

The Physicist. Show me results and insights from neuromodeling that demonstrate how the macroscopic behaviour of complex systems can be understood to be dependent

[1] The "Physicist" viewpoint is in addition to those found in (Churchland and Sejnowski 1992).

on a compact description of the system.
Non physicists do not know anything much about statistical mechanics and how to bridge the gap between a microscopic description and understanding the resulting macroscopic behaviour.

We do not proceed here to attempt to unify these differing viewpoints. Indeed, this diversity of viewpoints is, in our opinion, a healthy feature. Our own work has been most closely associated with the "computer scientist" and "physicist" viewpoints. As we see it, the problem of making artificial machines endowed with some of the functionality of biological organisms (*e.g.,*visual processing of information) is so complex, and the *a priori* space in which to search for possible solutions so vast, that we must turn to those biological organisms and their specific functionality if we are to succeed in our goal. In so doing, we need to familiarise ourselves with relevant mathematical frameworks since, ultimately, we wish to find a mathematical description of the procedure. In spirit this is similar to David Marr's view of artificial intelligence, which relates artificial intelligence to coding specific biological functions. (See his article in Boden's Book and others therein for an introduction to different viewpoints on artificial intelligence and related philosophical issues). In concentrating on making an "artificial neural network" which is capable of information processing in a manner similar to the brain, there are some observations about the brain which highlight some of the differences to conventional computation (see Hertz, *etal*):

- It is robust and fault tolerant. Nerve cells die every day without affecting its performance significantly.
- It is flexible. It can easily adjust to a new environment by "learning" – it does not have to be programmed in a standard computer language.
- It can deal with information that is fuzzy, probabilistic, noisy, or inconsistent.
- It is highly parallel.
- It is small, compact, and dissipates very little power.

Ultimately, we would like to be able to make a machine that can perform certain information processing tasks such as face or speech recognition that humans can do with consummate ease. The starting point here is that conventional approaches based on Artificial Intelligence have reached an impasse, since the task of formally specifying a task such as face recognition is either unclear or too complex to be handled in a conventional way.
We split our review into the following main sections. The first, section (2) deals with the neural biological background that ultimately motivated the field. We then show how this motivated some of the earliest artificial neural network models in section (3).) The generalisation of such early models leads us into the realm of statistical physics in section (4). This subfield of neural networks focuses on the emergence of macroscopic behaviour from the detailed microscopic desciptions of neural networks. A particularly useful area of research has been the development of neural networks as advanced statistical models. In section (5) we review some of the progress that has been made in the machine learning community under the general banner of neural networks and in particular those developments in learning non-linear mappings parameterised as perceptrons. This section contains more immediate practical issues of neural networks and related methods. There is a wealth

of material on this topic, and an introduction can be found in the references. More general and recent discussions on how to train models such as neural networks are treated in section (6), where we compare frequentist and Bayesian approaches. How neural networks have found commercial success in applications is outlined in section (7). Finally, we conclude in section (8) with a summary and outlook on where artificial neural networks might be heading in the near future.

2 Neurobiology

2.1 Neurons

Neurons are the basic structural components of the brain[2]. A neuron is an individual cell, specialised by architectural features that enable fast voltage changes across its membrane as well as voltage changes across neighbouring membranes. Brains are assemblies of such cells, and while an individual neuron does not see or reason or remember, brains do. How can we get from ion movement across cell membranes to memory or perception in brains. What is the nature of neuron-neuron connectivity and interactivity? What makes a clump of neurons a nervous system?Two ground-breaking discoveries in the nineteenth century established the foundations for a science of nervous systems. (1) Macro effects displayed by the nervous systems depend on individual cells, whose paradigm anatomical structure include both long tails (axons) for sending signals and treelike proliferation (dendrites) for receiving signals, see fig (1). (2) These cells are essentially electrical devices; their basic business is to receive and transmit signals by causing and responding to electric current. Within the last few decades, an enormous amount has been learned about neurons: about their electrophysiology, microanatomy, connectivity and development. If we know so much about the fundamental microscopic aspects of the brain, neurons, surely we also have a good understanding of macroscopic aspects such as how the visual or motor system works. In fact, we do not. It could be that we simply do not yet understand

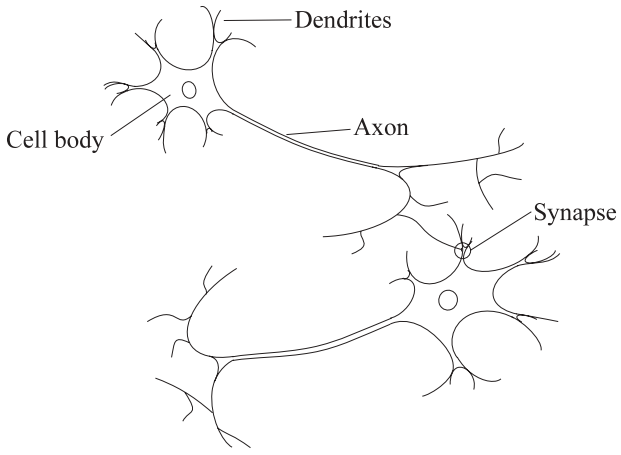


Figure 1: A schematic representation of two neurons and their connection point at the synaptic junction. The cell body receives (electrical) input along its dendrites. Provided that this combined input is high enough, a spike or pulse is transmitted along the axon, branching out to many (typically of the order of a thousand) synaptic junctions. These signals are then received by an afferent neuron along its dendrite.

[2]Much of this overview of neurobiology is taken from (Churchland and Sejnowski 1992) and (Hertz, Krogh, and Palmer 1991).

in enough detail how neurons work – ultimately, proponents of this bottom-up research approach contest that we will be able to understand large scale phenomena in the brain. However, the main argument of some theoretical neurobiologists is that, no matter what level of detail the individual neuronal aspects of the brain are understood, this is not sufficient to explain the complex large scale properties of the brain, such as visual awareness. Such researchers contend that such complex behaviours can only be realised when such neurons and coupled together, producing a dynamic, highly non-linear information processing system, the power and properties of which cannot be understood merely by the study of neurons in isolation. This approach is central to the field of computational neuroscience. This field aims for biological realism in computational models of neural networks which may, however, study at times relatively simple models to see if they are sufficient at qualitatively explaining emergent biological phenomena.

2.2 Simple Neuron Models

The brain is composed of about 10^{11} neurons of many different types, a common class of which has the form depicted in fig (1). Tree like networks of nerve fibre called dendrites are connected to the cell body or soma, where the cell nucleus is located. Extending from the cell body is a single long fibre called the axon, which branches into strands and substrands. At the ends of these are the transmitting ends of the synaptic junctions, or synapses to other neurons. The receiving ends of these junctions on other cells can be found both on the dendrites and on the cell bodies themselves. The axon of a typical neuron makes a few thousand synapses with other neurons.

The transmission of a signal from one cell to another at a synapse is a complex chemical process in which specific transmitter substances are released from the sending side of the junction. The effect is to raise or lower the electrical potential inside the body of the receiving cell. If the cell body potential of a neuron (after receiving inputs from its neighbours) reaches a threshold, a pulse or action potential of fixed strength and duration is fired, which is propagated along the axonal arborization to synaptic junctions of other cells. After firing, the cell has to wait for a time called the refractory period before it can fire again.

McCulloch and Pitts in 1943 proposed a simple model of a neuron as a binary threshold unit. Specifically, the model neuron computes a weighted sum of its inputs from other units, and outputs a one or a zero according to whether this sum is above or below a certain threshold:

$$n_i(t+1) = \Theta\left(\sum_j w_{ij} n_j(t) - \mu_i\right).$$

Here n_i is either 1 or 0, and represents the state of neuron i firing or no firing respectively at time t. Θ is the step function, $\Theta(x) = 1$ if $x \geq 0$ and $\Theta(x) = 0$ otherwise.

The weight w_{ij} represents the strength of the synapse connecting neuron j to neuron i. It can be positive or negative corresponding to an excitatory or inhibitory synapse respectively.

Though individually simple, a collection of McCulloch-Pitts neurons forms a computationally powerful device. Indeed, a synchronous assembly of (sufficiently many) such neurons is capable of universal computation, programmable by choosing weights w_{ij}, and can thus perform any computation that an ordinary digital computer can do.

This simple description differs from real neurons in some fundamental ways.

- Real neurons respond to their input in a continuous way. However, the non-linear relationship between the input and the output is a universal feature. A working hypothesis is that nonlinearity is essential, though not its specific form.
- Real neurons perform a nonlinear summation of their inputs.
- A real neuron produces a sequence of pulses, not a simple output level. Representing the firing rate by a single number n_i, even if continuous, ignores the possibility that pulse phase, the timing of individual "spikes", not just the rate, encodes a significant amount of relevant information.
- The amount of transmitter substance released at a synapse may vary unpredictably.

A simple generalisation of the McCulloch-Pitts neuron which includes some of these features is

$$n_i(t+1) = g\left(\sum_j w_{ij} n_j(t) - \mu_i\right)$$

where g is a continuous function. To take into account some of the stochastic effects, we could alternatively consider

$$p(n_i(t+1) = 1) = g\left(\sum_j w_{ij} n_j(t) - \mu_i\right) \tag{1}$$

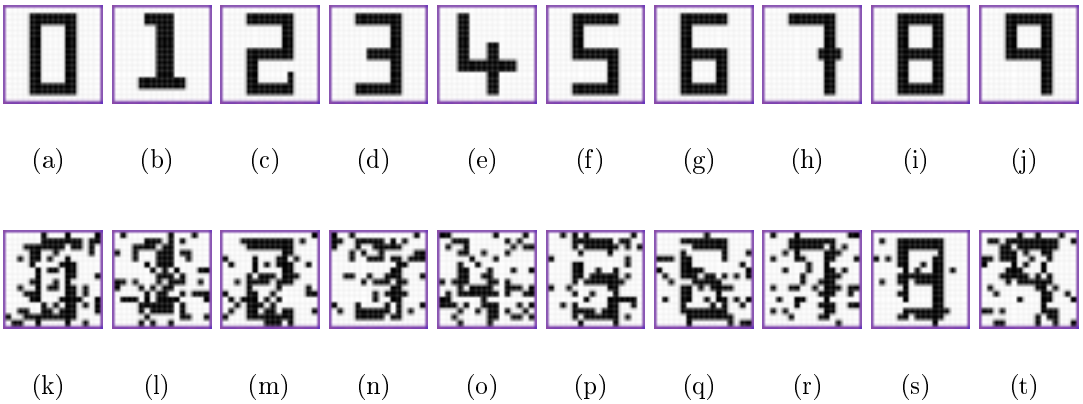
where g is a function between 0 and 1, so that (1) represents the probability that neuron i fires in a unit time interval.
In most applications of (classical) neural networks, the former interpretation of neurons is applied. That is, the output of each network is a deterministic (nonlinear) function of its inputs. This is then fed successively into other neurons, and the process repeated. We shall mainly deal with this approach in section (5). The stochastic case, in which we consider the output as representing the probability that the neuron fires, is more closely related to systems in statistical physics, and we shall deal with this more closely in section (4). The tools of statistical mechanics may be applied to analysing the properties of both kinds of deterministic and non-deterministic systems. Interestingly, the stochastic model of neurons is a special case of a wider class of statistical models known as graphical models. Graphical models were introduced in response to the failure of traditional expert systems to cope with uncertainty, and is currently a hot research area.

3 The Hopfield Network

Hopfield networks are early models based on the simple McCulloch-Pitts type neuron. It is conventional here to deal rather with $+1$ (firing) or -1 (quiescent) variables, rather than the original 0/1 variables, so that explicitly, the model is[3]

$$s_i = \mathrm{sgn}\left(\sum_j w_{ij} s_j\right) \tag{2}$$

[3]More generally, we can consider the case of thresholds $s_i = \mathrm{sgn}\left(\sum_j w_{ij} s_j - \mu_i\right)$. For expositional clarity, however, we neglect thresholds here.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j)

(k) (l) (m) (n) (o) (p) (q) (r) (s) (t)

Figure 2: (a) to (j) The 10 patterns stored by the Hopfield network. (k) to (t) Noise corrupted versions of the patterns successfully recognised by the network. Below each stored digit is an example of a noisy version of that digit, recognised by the network. That is, when initialised in the noisy state, the Hopfield Network converged to the correct pattern.

where $\text{sgn}(x) = 1$ if $x \geq 0$ and $\text{sgn}(x) = -1$ otherwise.
An interesting question is whether or not the neurons, under the dynamics (2) converge, or whether they oscillate in an arbitrary fashion ad infinitum. Consider the following function

$$H = -\frac{1}{2}\sum_{ij} w_{ij} s_i s_j.$$

We claim that under the dynamics (2), this "energy" function decreases in time, provided that the weights are symmetric, $w_{ij} = w_{ji}$ and $w_{ii} = 0$.
For symmetric connections, we can write

$$H = C - \sum_{(ij)} w_{ij} s_i s_j$$

where (ij) means the sum over all distinct ij pairs. Let s_i' be the new value of neuron s_i. Clearly, if this is the same as the old value s_i, then the energy function does not change. Otherwise $s_i' = -s_i$, so that the change in energy is

$$H' - H = -\sum_{j \neq i} w_{ij} s_i' s_j + \sum_{j \neq i} w_{ij} s_i s_j = 2s_i \sum_{j \neq i} w_{ij} s_j = 2s_i \sum_j w_{ij} s_j - 2w_{ii}.$$

The first term in the above equation is negative from (2), and the second is zero by prescription, so that the energy necessarily decreases.

3.1 The Hebb rule

Given that there exists an energy function (Lyapunov function) which decreases with respect to the dynamics, we know that there must be some stable states of this dynamical system. How can we construct the attractors of this system to be patterns u that we wish to recall? In 1949 Donald Hebb suggested that synaptic strengths in the brain change

proportional to the correlation between the firing of the pre and post synaptic neurons. The following prescription for the weights is one way to achieve this

$$w_{ij} = \frac{1}{N} \sum_{\mu=1}^{P} u_i^\mu u_j^\mu$$

where μ labels the patterns that we wish to store.
To show that this gives rise to a stable pattern, consider what happens when the network is in the state u_i^ν. The pattern is stable if, for all i,

$$\text{sgn}\left(\sum_j w_{ij} u_j^\nu\right) = u_i^\nu.$$

Using the prescription for the Hebb rule, we get

$$\sum_j w_{ij} u_j^\nu = \sum_j \sum_\mu u_i^\mu u_j^\mu u_j^\nu = u_i^\nu + \frac{1}{N} \sum_{\mu \neq \nu} u_i^\mu \sum_j u_j^\mu u_j^\nu.$$

Provided that the second term is small enough, then the pattern will still be stable since this does not override the first term when we take the sign of this expression. A very rough argument for randomly generated patterns shows that such patterns will be stable: The second term contains a sum over roughly NP binary variables $u_i^\mu u_j^\mu u_j^\nu$ with mean zero and variance $\mathcal{O}(1)$. The typical magnitude of such a sum is $\sqrt{NP}$, which is less than N provided that $P \leq N$. This suggests that as long as the number of patterns is of the order of the number of neurons in the system, then the patterns will be stable. In fact, a more rigorous analysis shows that the number of patterns that can usefully be stored (that is, with non negligible basins of attraction) is approximately $P = 0.138N$.
An example application of Hopfield networks is given in fig (2), where noise corrupted versions of digits are recognised correctly by the model. What is interesting about this approach is not so much that this task has been solved (since there are many other ways to find good solutions to this classification problem), rather that it has been solved in a manner which has some biological relevance, and incorporates some of the desires that we set out for an artificial "biological" information processing machine.

4 Statistical Physics

4.1 The Boltzmann Machine

The Boltzmann machine is an extension of the Hopfield network in two ways. (1) It is a stochastic network, in which a probability distribution over the possible states of the network is defined. (2) Hidden units are employed to increase the computational complexity that such machines can perform. Specifically, the probability that the neurons (or "nodes") are in state $\mathbf{s} = (s_1, \ldots, s_n)$ is

$$p(\mathbf{s}) = \frac{1}{Z} \exp\left(-\frac{1}{2T} \sum_{ij} w_{ij} s_i s_j\right) \tag{3}$$

where $Z = \sum_{\{s_i=+1,-1\}} \exp\left(-\frac{1}{2T} \sum_{ij} w_{ij} s_i s_j\right)$ is a normalising constant. Equation (3) is of the form $\exp(-H/T)$ where H is the Hopfield energy (albeit with additional hidden states)

in which T plays the role of "temperature". The lower T is, the more certain it becomes that the system inhabits only low energy states. If T is high, all states become equally probable, and there is no perceived order. The Boltzmann machine is a density estimator, and a relevant question is therefore, what are the typical states that this distribution inhabits? Consider the problem of generating a set of typical states $\mathbf{s}^1, \mathbf{s}^2, \ldots$ from the distribution $p(\mathbf{s})$. A simple Monte Carlo sampling algorithm, Gibbs sampling, considers updating a single neuron, conditional on the other neurons,

$$p\left(s_i | s_1, \ldots s_{i-1}, s_{i+1}, \ldots s_n\right). \tag{4}$$

In this case, this gives

$$p\left(s_i | s_1, \ldots s_{i-1}, s_{i+1}, \ldots s_n\right) \propto e^{-\frac{1}{T} s_i \sum_j w_{ij} s_j} \tag{5}$$

which can be conveniently written as

$$p\left(s_i | s_1, \ldots s_{i-1}, s_{i+1}, \ldots s_n\right) = \sigma\left(s_i \sum_j w_{ij} s_j / T\right) \tag{6}$$

where the sigmoidal function $\sigma(x) = e^x / (e^x + e^{-x})$. In the limit $T \to 0$, $\sigma(x) \to \Theta(x)$, and the sampling equation (6) becomes the Hopfield update rule. In this sense, Hopfield networks are a special case of Boltzmann machines.

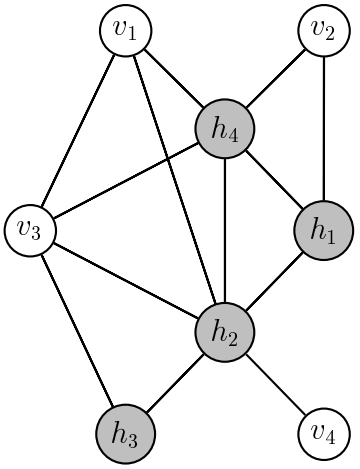
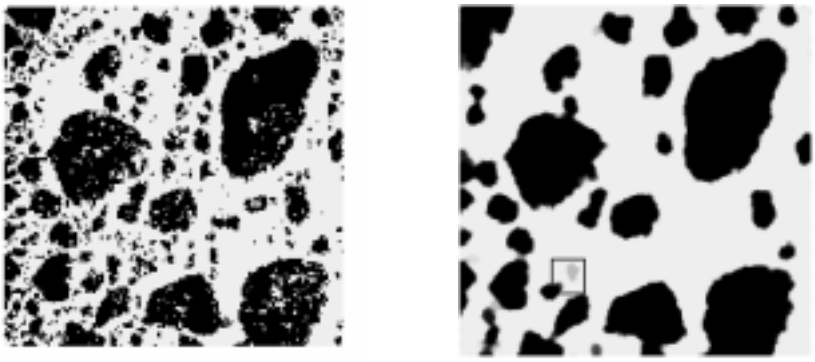


Figure 3: A Boltzmann machine. The nodes are split into two groups, hidden (light gray), and visible (white). The hidden units increase the complexity of the visible unit distribution through marginalisation.

A further departure from the Hopfield network comes in the realisation that "hidden" units $\mathbf{s}_h$ can be employed to increase the computational power (complexity of the network distribution) on the "visible" nodes $\mathbf{s}_v$. We therefore define a joint distribution $p(\mathbf{s}_h, \mathbf{s}_v)$ on both the hidden and visible units (here $\mathbf{s}_h$ and $\mathbf{s}_v$ denote sets of nodes). In considering how to find appropriate parameters for specifying the distribution on the visible units, we need to marginalise this joint distribution over the hidden units,

$$p(\mathbf{s}_v) = \sum_{\mathbf{s}_h} p(\mathbf{s}_h, \mathbf{s}_v).$$

The difficulty in using the Boltzmann machine stems from the difficulty in carrying out this summation over the hidden states. This is essential since, if we are to find appropriate weights w_{ij} to learn a set of visible patterns, we need to compare the likelihood of a (visible) pattern given the current set of weights. For general connections w_{ij}, the time taken to compute this marginal is typically exponential in the number of hidden units. Despite recent advances in approximations to this summation, this still remains a difficult issue holding back the wider application of Boltzmann machines.



(a) Original Noisy Image (b) Restored Image

Figure 4: Image restoration using a Boltzmann machine. The noise in the original image (a) is cleaned up to to give the restoration, (b).

4.2 Statistical Mechanics of Ising Systems

The Boltzmann machine corresponds to a simple model of a magnetic system, known in the physics literature as the Ising model. This model has been studied intensively since its introduction in 1925. Although the description of the "spin 1/2" atoms in this model is very simple, the model displays a qualitative behaviour similar to the case of real magnetic systems. In the Ising model, the connection weights w_{ij} are non-zero only for nearest neighbour interactions, and their sizes are drawn from a fixed zero mean, unit variance distribution. When the temperature T is high, the interaction between spins is weak, and their is little order apparent – all spins point in random directions. As the temperature goes below a critical point, there is a phase transition, and the system can behave as a magnet. What is interesting about this phenomenon is that, despite the microscopic complexity of this type of system, the macroscopic behaviour is determined by a small number of variables (order parameters) such as the temperature and the distribution from which the inter-spin connections are drawn. This indicates that a similar simplified description of neuronal behaviour may be possible, so that we can describe the macroscopic behaviour of certain brain functions in terms of a relatively small set of variables. One important difference, however, between such simple magnetic models and more realistic models of biological is that there are invariances in the magnetic system, such as translation invariance, which are unlikely to be present in biological systems. Some caution needs to be maintained then in interpreting the significance of the connection between analyses of simple physical models and their potential applications to more realistic biological systems.

4.3 Markov Random Fields

The Boltzmann machine is an example of a Markov Random Field, which has found application in image restoration. The strengths of nearest neighbour interactions can be used to encode our prior beliefs about what kinds of images are likely to occur (just as in the Hopfield model, the weights can be used to store patterns). Given a corrupted image, and an assumption about the corruption process, we can recover an approximation of the

original image by finding which visible is the most likely state to have given rise to such a corrupted observation. In fig (4) we see an example application, in which an original, noisy image, fig(4a) is restored using the Boltzmann machine to the cleaner image fig(4b).

5 Perceptrons

A particularly fruitful area of research has been in developing neural networks as advanced statistical models. Hopfield networks and Boltzmann machines are classified as *unsupervised* learning models. In *supervised* learning, it is more natural to consider that for each (input) pattern, there is an associated output pattern (specified by the supervisor), and it is this input-output relationship that needs to be learned. There are many instances of such a learning problem – for example, whether or not a person is suitable for a credit loan. There are many advanced models currently available for such tasks, many of which have their roots in simpler models, historically known as perceptrons. Indeed, much of the initial publicity was generated as a consequence of the performance of perceptrons on supervised learning tasks.

5.1 The Perceptron

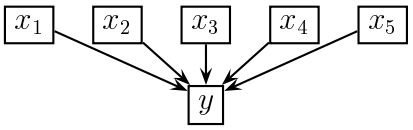


Figure 5: A simple perceptron. We use square boxes to emphasise the deterministic nature of the network.

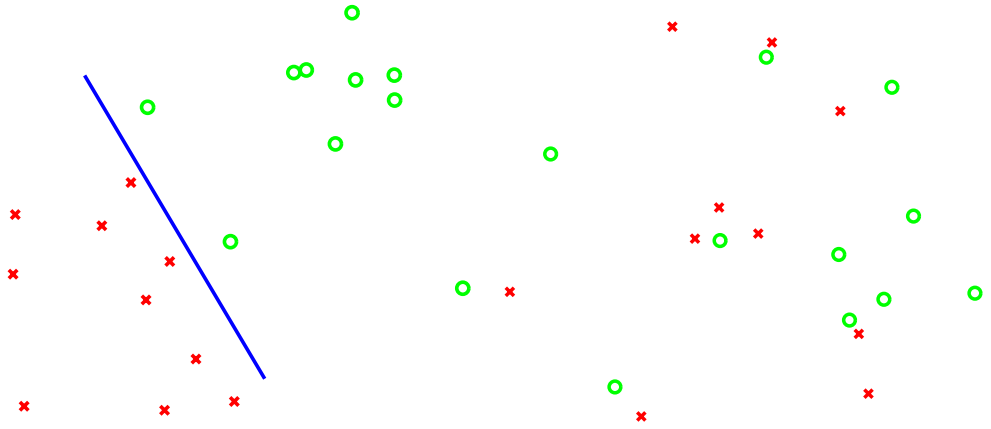
The perceptron is essentially just a single neuron like unit that computes a non-linear function y of its inputs x,

$$y = g\left(\sum_j w_j x_j + \mu\right)$$

where the weights $\mathbf{w}$ encode the mapping that this neuron performs. Graphically, this is represented in fig (5). We can consider the case of several outputs as follows:

$$y_i = g\left(\sum_j w_{ij} x_j + \mu_i\right)$$

so that a perceptron can be used to model an input-output mapping $\mathbf{x} \to \mathbf{y}$. Coupled with an algorithm for finding suitable weights, we can use a perceptron for regression. Of course, the possible mappings the perceptron encodes is rather restricted, so we cannot hope to model all kinds of complex input-output mappings successfully. For example, consider the case in which $g(x) = \Theta(x)$ – that is, the output is a binary valued function. In this case, we can use the perceptron for binary classification. With a single output we can then classify an input $\mathbf{x}$ as belonging to one of two possible classes. Looking at the perceptron, (8), we see that we will classify the input as being in class 1 if $\sum_j w_j x_j + \mu \geq 0$, and as in the other class if $\sum_j w_j x_j + \mu < 0$. Mathematically speaking, the decision boundary then forms a hyperplane in the $\mathbf{x}$ space, and which class we associate with a datapoint $\mathbf{x}$ depends on which side of the hyperplane this datapoint lies, see fig (6). A simple learning rule known as the perceptron learning rule can be used to find an appropriate set of weights $\mathbf{w}$ that minimises the number of errors that the perceptron makes on a given set of training datapoints.



(a) A linearly separable problem (b) A non-linearly separable problem

Figure 6: Linear separability: The data in (a) can be classified correctly using a hyperplane classifier such as the simple perceptron, and the data is termed linearly separable. This is not the case in (b) so that a simple perceptron cannot correctly learn to classify this data without error.

5.2 Multilayer Perceptrons

If the data that we are modeling is not linearly separable, we have a problem, since we certainly cannot model this mapping using the simple perceptron. Similarly, for the case of regression, the class of function mappings that our perceptron forms is rather limited, and only the simplest regression input-output mappings will be able to be modelled correctly with with a simple perceptron. These observations were pointed out in 1969 by Minsky and Papert and depressed the research in this area for several years. A solution to this perceived problem was eventually found, which included "hidden" layers in the perceptron, thus increasing the complexity of the mapping. One such example is given in fig (5.1), in which the inputs are mapped by a non-linear function into the first layer outputs. In turn, these are then fed into subsequent layers, effectively forming new inputs for the layers below. Generally, the more layers that there are in this process, the complexity of the class of functions that such MLPs can model increases. However, it can be shown that, provided that there are sufficiently many

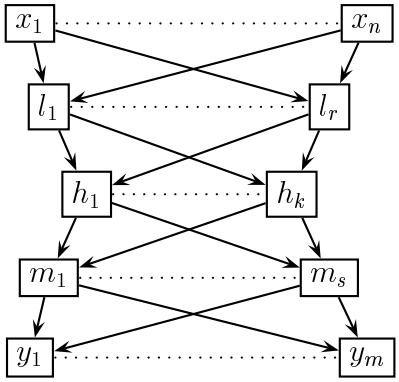


Figure 7: A multilayer perceptron (MLP) with multiple hidden layers, modeling the input output mapping $\mathbf{x} \rightarrow \mathbf{y}$. This is a more powerful model than the single hidden layer, simple perceptron. We used here boxes to denote the fact that the nodes compute a deterministic function of their inputs, as opposed tot the stochastic nature of graphical models such as the Boltzmann machine in fig (3)

units, a single hidden layer MLP can model an arbitrarily complex input-output regression function.

6 Training Neural Networks

6.1 Neural Networks as Advanced Statistical Tools

6.1.1 Training multi-layered perceptrons

To a statistician, neural networks are a class of non-linear (adaptive basis function) models. This is a very powerful class of models, and neural networks have some nice properties. Specifically, multi-layered perceptrons take the form[4]

$$f(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{K} v_i \sigma(\mathbf{w}_i \cdot \mathbf{x}) \tag{9}$$

where $\sigma(h)$ is a sigmoidal function (a monotonically increasing, bounded function, such as $1/(1+e^h)$), and $\mathbf{w}_i$ and v_i are parameters of the model, the combination of which is denoted $\mathbf{w}$. Training neural networks basically means statistical inference of the parameters $\mathbf{w}$.
Let us consider the case of regression, for convenience with a single output variable y. Given a set of input-output pairs, $D = \{\mathbf{x}^\mu, y^\mu\}$ with$\mu = 1 \ldots P$, how can we find appropriate weights $\mathbf{w}$ that minimise the error that the network makes in fitting this function? In neural-network terminology, we would define an an "energy" function that measures the errors that the network makes, and then try to minimise this function with respect to the weights $\mathbf{w}$. For example, a suitable choice of energy or error function might be

$$E\left(\mathbf{w}|D\right) = \sum_{\mu} \left(y^\mu - f\left(\mathbf{x}^\mu, \mathbf{w}\right)\right)^2 \tag{10}$$

where $f\left(x^\mu, \mathbf{w}\right)$ is the output of the network for input x^μ, given that the parameters describing the network are $\mathbf{w}$. We can train this network by any standard (non-linear) optimisation algorithm, such as conjugate gradient descent or Levenberg-Marquardt. In computing the gradient of the error function, naively it appears that we need of the order of PW^2 operations (if W is the number of parameters in the model and P is the number of training patterns), since computing the output of the network involves roughly W summations for each of the P patterns, and the gradient is a W-dimensional vector. The essence of the famous backpropagation procedure is that the gradient can instead by computed in order PW operations. If the training set is very large, standard computation of the gradient over all training patterns is both time-consuming and sensitive to round-off errors. In that case, "on-line learning", with weight updates based on the gradient for individual patterns, offers an alternative.

6.1.2 Statistical inference and overfitting

In principle, the problem of training neural networks is equivalent to the general statistical problem of fitting models to data. It is fair to say that some initial cavalier attitudes

[4]We here consider only a single hidden layer, and biases are neglected since they can be incorporated more naturally into the framework by augmenting the input space with an extra dimension, with constant, unit input

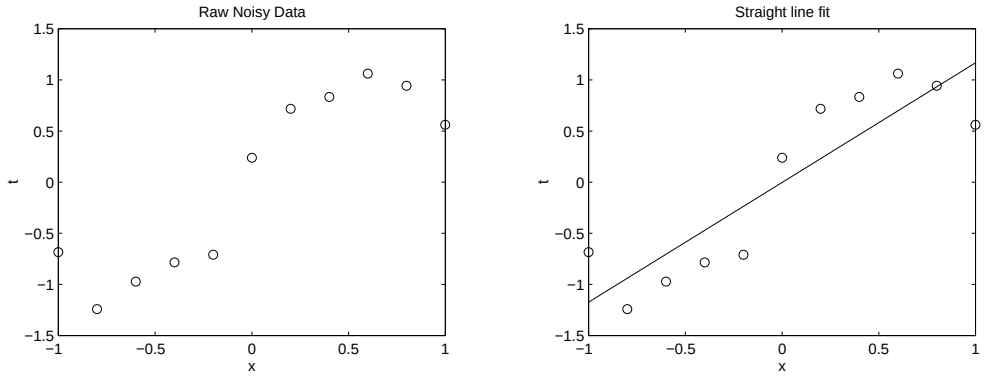


Figure 8: Curve fitting. On the left, we see an example of data, and our task is to fit a curve from which we believe that this data has been drawn. The straight line fit on the right is one such plausible fit. However, it is too simple and "underfits" the data.

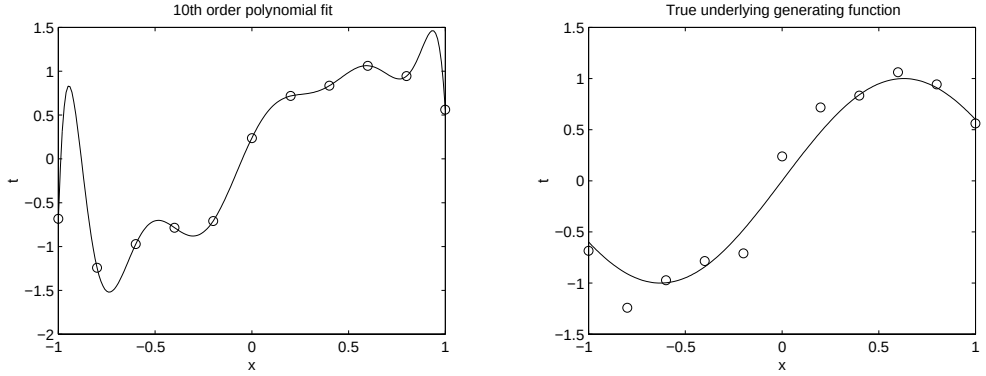


Figure 9: For the same data as in fig (8), we see that we can fit a curve exactly through the data using a 10^{th} order polynomial. However, this "overfits" the data, leading to poor generalisation in some regions. The true underlying function from which this data was generated is presented on the right, where the output has been corrupted with additive Gaussian noise of variance 0.2^2.

of neural net practitioners in wielding enormously complex non-linear models have been correctly sobered by the statistics community and that, conversely, the application of non-linearity in statistics has been boosted significantly by embracing of neural network like techniques. Basically, there are two different paradigms for the inference of parameters from data: the classical frequentist and the Bayesian approach. The classical approach is most similar to the way early neural-network researchers described training neural networks. The Bayesian approach is becoming more and more popular and may, in the end, appear to be the most fruitful paradigm. See Loredo's article for an excellent discussion and practical examples of the Bayesian versus classical frequentist debate.

One of the main problems when fitting complex non-linear models to data is how to prevent "overfitting", or, more generally, how to select the model that not only fits the data, but also generalises well to new data. The concept of overfitting is visualized in fig (8) and fig (9). The data, output y versus input x, is shown in fig (8) on the lefthand side. A simple linear perceptron would yield the fit on the righthand side. A much more complex

model, *e.g.*,a 10^{th} order polynomial but also a neural network with a large number of hidden units, would yield a perfect on the data, as shown in fig (9) on the lefthand side. However, considering the true underlying function displayed on the righthand side, it is clear that the straight line is too simplistic and underfits the data, whereas the 10^{th} order polynomial is too complex and overfits the data. In the subsequent sections, we will discuss the frequentist and Bayesian approaches to find the "right" model.

6.2 The Classical Approach

6.2.1 Regularisation

In the classical approach, the goal is to find the "optimal" $\mathbf{w}^*$ that minimises an error function $E(\mathbf{w})$. A typical example of such an error function is the sum-squared error in (10). The "best" model given an input x is then the function $f(x, \mathbf{w}_*)$. An example is given in fig (10). Given the data points, the fit found by the neural network seems quite reasonable.
In practice, however, direct optimisation of neural networks is prone to the overfitting problem explained above. The standard, so-called *regularisation* approach is to add a penalty term to the original error (10), for example, a term which penalises large weights,

$$E_{reg}(\mathbf{w}|D) = E(\mathbf{w}|D) + \lambda \mathbf{w} \cdot \mathbf{w}. \tag{11}$$

The larger λ is, the smoother (more regularised) the function will be. The question however is, how do we set λ?

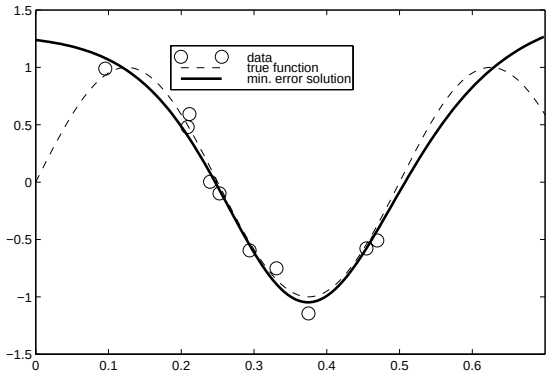


Figure 10: The minimum training error solution neural network fit (solid line) to data. The true underlying function is given by the dashed curve.

6.2.2 Model choice

The setting of regularisation parameters such as λ above is an example of the central issue of model choice. A solution to this difficulty is to (randomly) split the data into a *training* and *test* set, $D = \{D_{train}, D_{test}\}$. Two values of λ (two models), can be used in the fit to the training data using the (regularized) least squares procedure. The two resulting $\mathbf{w}_*$ parameters, one for each λ value, can be tested on the test data, to give two test errors,

$$E(\mathbf{w}_*|D_{test}) = \sum_{(x,t)\in D_{test}} \left(f(x, \mathbf{w}_*) - t\right)^2 .$$

That model (λ) with the lower test error is then preferred. The generic term for this procedure is "cross-validation" and is widely used, especially if little is known about the problem at hand, and the only criterion is for a model with a low test error - in other words, good *generalisation*. For high complexity (here, low λ), the training data is overfitted, and the test error is high. As λ is increased, underfitting occurs and again, the test error is high. The

optimal complexity (λ) occurs somewhere in between, see fig (11). This trade-off, between model complexity and data fit, is sometimes referred to as the "bias/variance dilemma".

6.2.3 Training ensembles of networks

Cross-validation is a typical frequentist procedure. It requires the subdivision of the original data in a training and a test set. In principle, this subdivision is rather arbitrary and usually done at random. But then, one might as well repeat the procedure and generate different networks, each trained on a different subset of training data. This idea, to train several networks instead of just one, has become very popular lately, and reduces the instability of the final solution. A popular frequentist method for subdividing the data in training and test patterns is bootstrapping. Many algorithms have been proposed on how to combine the results of the separate networks into a single answer and for computing error bars that indicate the reliability of this answer.

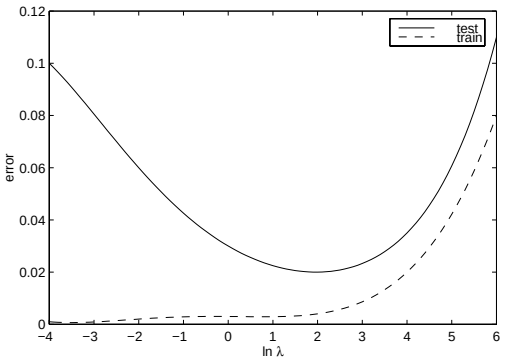


Figure 11: The training/test error trade-off. When the model is too flexible (small λ), overfitting occurs (too low training error), and generalisation is poor (high test error). If model flexibility is over penalised (large λ), the data cannot be fitted, and the model generalises poorly. In between lies a λ giving better generalisation.

6.3 The Bayesian Framework

6.3.1 The Bayesian philosophy

The Bayesian framework provides an alternative to the frequentist procedures and is rapidly gaining popularity. It is based on a quite different philosophy.

- All models are *wrong*! However, we can subjectively quantify our beliefs about which models are, *a priori* (what we know about the model before we have seen the data), more suitable. The statistical problem is then to combine these beliefs with the data in a consistent way to yield, a posteriori (what we know about the model after we have seen the data), beliefs in order that the suitability of competing models may be quantified.

- It is important to understand if the inadequacies of the method are poor model assumptions or if there are (computational) difficulties in applying the statistical model fitting method. If possible, such application difficulties should be minimised, so that one can concentrate on the model assumptions.

- Inference without assumptions is impossible. The Bayesian approach forces the modeller to make explicit all assumptions about the model, rather than burying them in an ad hoc procedure. However, this elegant framework may, in some cases, be compromised by computational difficulties.

6.3.2 Prior and posterior

Applied to the task of curve fitting, the Bayesian machinery more or less works as follows. First we need to describe the class of functions (models) that we choose to fit the data. An example could be a neural network with a particular number of hidden units. These model assumption are usually summarized in the hypothesis H. To find a "good" curve, we further need beliefs about the smoothness of the "clean" underlying function and the level and nature of the noise in any possible corruption process. For convenience we consider the case in which the noise process is known and we only have a parameter λ specifying our smoothness belief. This belief is specified through the prior probability $p(\mathbf{w}|\lambda, H)$, *i.e.*,, $p(w|\lambda, H)$ quantifies our *a priori* belief in the appropriateness of the value of the parameter $\mathbf{w}$ given (hyper)parameter λ and other assumptions H. A typical choice is

$$p(w|\lambda, H) \propto \exp\left[-\frac{\lambda}{2}w^2\right]. \tag{12}$$

Our belief in what appropriate parameters $\mathbf{w}$ are should change when data D comes in. Exactly how follows from Bayes' rule:

$$p(\mathbf{w}|D, \lambda, H) = \frac{p(D|\mathbf{w}, \lambda, H)p(\mathbf{w}|\lambda, H)}{p(D|\lambda, H)}. \tag{13}$$

$p(D|\mathbf{w}, \lambda, H)$ is called the *likelihood* – given that the function with parameter $\mathbf{w}$ (from our class of functions H), is the correct underlying function generating the data, what is the probability that the particular data set is observed? Under the assumption of Gaussian noise (with unit standard deviation), the probability of a dataset D reads

$$p(D|\mathbf{w}, \lambda, H) = p(D|\mathbf{w}) \propto \prod_{\mu} \exp\left[-\frac{1}{2}(y^{\mu} - f(x^{\mu}, \mathbf{w}))^2\right].$$

It is easy to check that, for a particular λ, the Bayesian *maximum a posteriori* solution, the maximum of posterior probability (13), corresponds to the frequentist $\mathbf{w}^*$ minimizing the regularised error (11). In other words, the frequentists' regularisation can be interpreted as an (approximate) Bayesian approach.

6.3.3 Bayesian model selection

The Bayesian approach to model selection is completely different from the frequentist one. In the Bayesian framework, our belief in a particular model directly follows from the rules of probability. For example, to compare two parameter settings λ_1 and λ_2, we compute

$$\frac{p(\lambda_1|D, H)}{p(\lambda_2|D, H)} = \frac{p(D|\lambda_1, H)p(\lambda_1|H)}{p(D|\lambda_2, H)p(\lambda_2|H)}$$

where

$$p(D|\lambda, H) = \int d\mathbf{w} p(D|\mathbf{w})p(\mathbf{w}|\lambda, H).$$

$p(\lambda|H)$ is our prior belief in the appropriateness of setting λ. We can do this comparison at the level of the hyperparameters, but also higher levels, for example, to compare two sets of models H_1 and H_2.

For non-linear models, unfortunately, Bayesian model comparison is typically computationally intractable since this involves integrals over high dimensional spaces, and approximations need to be introduced.

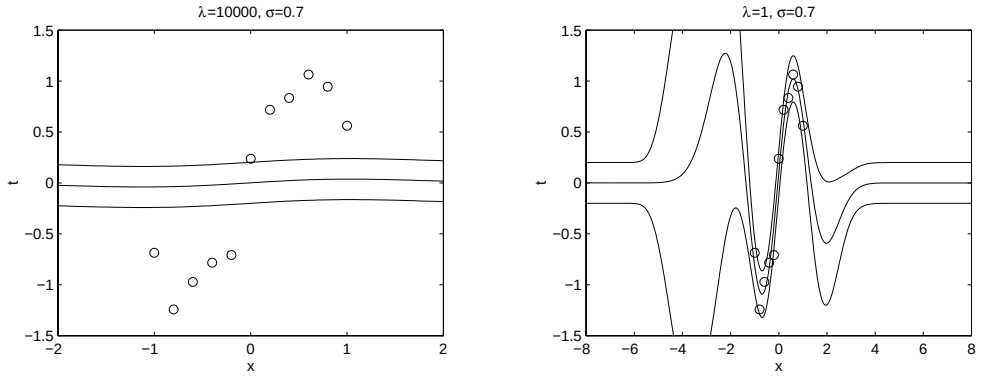


Figure 12: Mean prediction and one standard deviation error bars for noise level $\sigma = 0.7$ and regularization prior $\lambda = 10000$ (left) $\lambda = 1$ (right). It can be seen that larger λ encourages smoother functions.

6.3.4 An example

We illustrate the Bayesian approach on the curve fitting problem in fig (8) and fig (9). We consider the so-called generalised linear model $f(x, \mathbf{w}) = \mathbf{w} \cdot \boldsymbol{\phi}(x)$, where the basis functions $\boldsymbol{\phi}_i(x)$ are fixed Gaussian kernels. The model is linear in the parameters w_i, yet non-linear in the inputs x. We assume that the data is generated by a function from this function class with additive Gaussian noise with standard deviation σ. Our prior on the model parameters $\mathbf{w}$ is of the form (12). Both λ and σ are part of our model assumption. The posterior probability given the data follows from Bayes' rule as in (13).
Predictions at a new test point are given by the posterior average:

$$\bar{f}(x) = \int p(\mathbf{w}|D, \sigma, \lambda, H) f(x, \mathbf{w}) d\mathbf{w} \tag{14}$$

where M summarises σ, λ, and H. This integral is straightforward to carry out, resulting in an analytical expression, since it involves only Gaussian integration .An important advantage of the Bayesian procedure is that confidence intervals directly follow from the posterior posterior distribution of the parameters,

$$\delta^2 f(x) = \int p(\mathbf{w}|D, M) \left(f(\mathbf{w}, x) - \bar{f}(x) \right)^2 d\mathbf{w}. \tag{15}$$

The mean prediction and standard confidence intervals for two different settings of the regularization parameter are shown in fig (12).

6.3.5 Sampling Methods

The Bayesian framework is directly applicable also to non-linear neural networks. Unfortunately, in general, the posterior will not have a simple, Gaussian type form, even if we assume Gaussian noise and a Gaussian parameter prior. As such, averaging over the high dimensional posterior is in general intractable and approximations need to be made.
A theorem in statistics states that the posterior distribution of parameters will tend to a Gaussian as the number of data points increases (up to identifiable symmetries). A natural approximation is simply to a fit a Gaussian locally around a mode of the posterior and use that to approximate the averages. For smaller numbers of data points, this approximation

is not sufficiently accurate. An alternative is then to approximate the integral by a finite sample from the posterior:

$$\int p(\mathbf{w}|D) f(x, \mathbf{w}) d\mathbf{w} \approx \frac{1}{L} \sum_i f(x, \mathbf{w}^i)$$

where $\{\mathbf{w}^i, i = 1 \ldots L\}$ are samples from the posterior $p(\mathbf{w}|D)$.
In principle, this approach is exact in the limit of an infinite number of samples. Indeed, the number of samples required for an accurate evaluation can be very small (and is *independent* of the input dimension). However, generating these samples remains generally extremely difficult and sampling techniques are a subject of much current research, both in physics, statistics and neural networks.
Each of the samples $\mathbf{w}^i$ corresponds to a particular function, so that we can think of sampled functions from the posterior. For example, fig (13) shows some (function) samples from the posterior distribution. Although fundamentally different, the results of Bayesian sampling techniques can be quite similar to the frequentists' ensembles described in section (6.2.3).

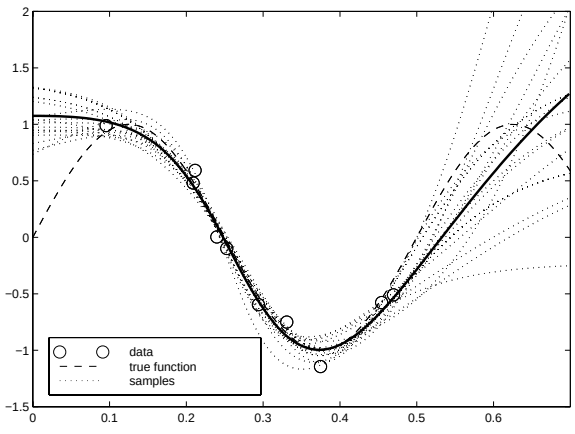


Figure 13: A set of (function) samples from the posterior distribution. From these we can readily find the mean prediction and associated error bars.

7 Neural Network Applications

7.1 General considerations

Most neural network applications derive from their learning capabilities. Mainly used are standard feedforward neural networks, either for classification or for regression. In these applications, the neural network serves as an alternative to standard statistical solutions. The underlying idea is that due their inherent nonlinearity, neural networks are better in modeling complex relationships than classical statistical methods. In the end, this need not always be the case. One of the reasons for the popularity of neural networks is that, using standard tools, a reasonable solution can be obtained in a reasonable amount of time. This gave neural networks the image of "second best" solutions.
This viewpoint is understandable when we consider standard neural packages, i.e., standalone software for horizontal applications across different domains. As explained before, a (feedforward) neural network is not principally different from standard statistical models. However, many of the available software packages do not treat neural networks as such. On the contrary, the thoroughness that surrounds classical statistical models (tools for computing confidence intervals, model and feature selection, outlier detection, and so on) is replaced with sloppiness under the presumption that neural networks are so powerful

that they can do without. Furthermore, the standard packages are often not flexible enough to handle all peculiarities of the problem to be solved. So, indeed, it is relatively straightforward to build simple neural applications with standard packages, but these are easily outperformed when more effort is put in.
An alternative, rapidly gaining ground, are neural toolboxes for statistical packages like SPSS, SAS, and Matlab. The threshold for usage is somewhat higher, the user has to buy and become acquainted with the statistical engine supporting the neural algorithms. The advantages over stand-alone neural packages are increasing flexibility and easy integration with statistical tools. This makes them very well suited for building tailored neural applications, examples of which will be given below. In these tailored applications, the neural machinery often only takes up a small but essential piece of the total solution.
Somewhere in between commercial stand-alone packages and toolboxes are the non-commercial neural packages mainly provided by research groups from universities. These often better represent the state-of-the-art with regard to learning algorithms and statistical embedding and, with availability of the source codes, are in principle adaptable. However, lack of documentation and support and obvious emphasis on the interests of the researchers themselves, often oppose user-friendliness.

7.2 Example applications

Applications of neural networks can be found in any domain where data is available to support decisions. The general term in this context is "data mining", also referred to as "knowledge discovery in databases". Neural networks are often quoted as *the* technology to be used, others being machine learning, clustering, statistics, and visualization techniques. Here we will give a short overview of neural data mining applications.

Marketing. Customer credit, billing, and purchases were some of the first business transactions to be automated with computers, yielding huge amounts of data available for mining in search for knowledge that can improve marketing results or lower marketing costs. A typical example is direct mailing. A test mailing is made to a small subset of customer. A feedforward neural network is used to model the response as a function of the characteristics of the customer. This model can then be used to determine who should be included in the subsequent mass mailing and which offers should be included. Other examples can be found in customer relationship management (enhance the revenues of existing customers by tuning marketing messages) and preventing customer retention (identifying customers who are likely to switch to competitors).

Retail and logistics. Neural networks are used for demand forecasting. In principle, these are standard time-series prediction problems in which neural networks have to compete with standard tools such as Box-Jenkins and ARMA. A nontrivial application has been developed for the prediction of single-copy newspaper sales. In this setting, predictions are needed on a daily basis for a huge set of individual outlets. By combining all outlets in a single neural network, the outlets can "learn from each other", e.g., by extracting typical demand features. In this setting, the neural architecture yields a clear benefit over standard approaches that treat all outlets individually.

Finance. Finance is *the* domain for success stories of the type "neural networks predict stock returns". Despite the fact that there may be successful solutions that temporarily work, including neural ones, the general feeling is that anything can be lucky. There are other problems in the financial domain that are better suited for a neural approach. Examples are the detection of fraudulent transactions with credit cards, portfolio optimization, predicting bankruptcies, and credit risk assessment. In most of these applications, neural networks are either used for time-series prediction or classification, their benefit over other tools being the capability of dealing with nonlinearities.

Manufacturing. The quality of a manufactured product often depends on the settings of many parameters. The exact relationship between these settings and the quality are often not well understood and too complex to describe with a physical or chemical model. Trained on examples yielding good and bad qualities, neural networks can provide a solution. Other applications of neural networks are in job shop scheduling and automatic inspection. In these control applications, neural networks are mainly used for function fitting to model (part of) the process one needs to control.

Health and medicine. Some of the applications in health and medicine resemble those in marketing and finance: detection of fraudulent insurance claims, risk assessment of clients, and so on. Other application relate to automatic diagnosis of diseases. It should be mentioned that in many medical applications, the surplus value of using neural networks over standard tools such as Cox survival analysis is often rather small. In many cases the databases are too small and too noisy to provide evidence of very complex relationships that would benefit from a neural approach.

Energy and utility. Prediction of energy demand is very relevant, both for large consumers who are often charged based on their peak energy usage, and for providers that have to anticipate upon extreme demands. In this context neural networks are used as nonlinear time-series predictors. A quite different kind of application in this area involves the detection of likely sites for gas and oil deposits. Based on all kinds of measurements at test drilling sites, neural networks are used to predict changes in the strata of rock, which relates to the presence of mineral deposits.

Summarising, there are indeed many (potential) applications of neural networks. Often it is not so much the technique that matters, but more the insight that appropriate use of available data can help to solve the problem. The exact technique becomes important for large-scale problems, where small improvements have major consequences. In general, neural machinery is most promising if there are nonlinear relationships between explanatory and response variables and sufficient data to find these. Especially the latter condition need not always be fulfilled, in which case the surplus value of neural networks over simpler techniques is limited. The trends are vertical applications, embedding in other statistical techniques, and combination of knowledge about the domain and available data. Most applications are based on classical frequentist statistics, although those using Bayesian methodology appear more and more.

8 Summary and Outlook

The field of neural networks has contributions from and makes contributions to many different areas. Although, ultimately, the motivation for much research has been biological, there are many areas in which artificial neural networks can and have been used successfully. More specifically, these include areas where the underlying process behind data generation is highly non-linear, and there now exists techniques that are able to give some confidence and insight into the performance of such models.

Two features of artificial neural networks stand out as being of particular importance – their non-linearity, and stochasticity (although this latter aspect is not always exploited in many applications). These properties can be used to define local computation units which, when coupled together suitably, can combine to produce extremely rich patterns of behaviour, whether these be dynamic, or static input-output relationships. One of the most import consequences of neural network research has been to bring the techniques and knowledge of artificial intelligence and statistics much closer together. Typically it was the case that problems in artificial intelligence were tackled from a formal specification aspect. On the other hand, statistics makes very loose formal specifications, and lets the data try to complete the model. Neural networks can be seen as a statistical approach to addressing problems in artificial intelligence, obfuscating the need for formal specifications of how the program works – just learn how to do it from looking at examples. For example, rather then formally specifying what constitutes the figure "2", a neural network can learn the (statistical) structure of "2"s by being asked to learn (find appropriate weights for) how to differentiate between "2"s and non-"2"s. This idea is especially powerful in the many human computer interaction applications where formally specifying, for example, what constitutes an individuals facial characteristics that differentiate them from others, is extremely difficult.
Recently graphical models have been popular and are, in a sense, a generalisation of stochastic neural networks. However, computing with these models is typically formally intractable (as in the Boltzmann machine), and researchers are currently looking into ways to find good approximation techniques, many derived from statistical mechanics and disordered systems.
Other more recent approaches that aim to find good statistical methods in artificial intelligence applications are support vector machines and Gaussian processes, although neither of these approaches retain many of the properties desirable of an artificial biologically inspired system.

Neurobiology continues to yield fascinating insights into the workings of biological brains, and it will be interesting to see how such developments impact on the problem of making artificial machines capable of tasks such as visual awareness. Whilst the current state-of-the-artificial-art in this respect remains primitive, we believe that the fusion of disciplines that neural networks encourage, will ultimately lead to a satisfactory result.

References

Anderson, J. and E. Rosenfeld (Eds.) (1988). *Neurocomputing: Foundations of Research*. Cambridge: MIT press.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Oxford University Press.

Castillo, E., J. M. Gutierrez, and A. S. Hadi (1997). *Expert Systems and Probabilistic Network Models*. Springer.

Churchland, P. S. and T. J. Sejnowski (1992). *The Computational Brain*. MIT press.

Crick, F. H. C. (1994). *The Astonishing Hypothesis*. Touchstone.

Gilks, W. R., S. Richardson, and D. J. Spiegelhalter (1995). *Markov Chain Monte Carlo in Practice*. Chapman and Hall.

Hertz, J., A. Krogh, and G. Palmer (1991). *Introduction to the theory of Neural Computation*. Redwood City California: Addsion-Wesley.

Jordan, M. I., Z. Gharamani, T. S. Jaakola, and L. K. Saul (1998). An Introduction to Variational Methods for Graphical Models. In M. I. Jordan (Ed.), *Learning in Graphical Models*, pp. 105–161. Kluwer.

Loredo, T. (1990). From Laplace to Supernova SN 1987A: Bayesian Inference in Astrophysics. In P. Fougere (Ed.), *Maximum Entropy and Bayesian Methods*, pp. 81–142. Kluwer.

MacKay, D. (2000). *Information Theory, Inference and Learning Algorithms*. `http://wol.ra.phy.cam.ac.uk/mackay/`.