Neural Networks in Accounting and Finance Research

Duarte Trigueiros

ISCTE,
Av. Forças Armadas, 1600 Lisbon, Portugal
Phone: +351 (1) 793 50 00, Fax: +351 (1) 796 47 10,
E-Mail: dmt@iscte.pt

Acknowledgements and Quotations. We acknowledge the support of Professor Luis B. Almeida at INESC, Portugal. This text can be referred to as a Report of the Department of Business Studies of ISCTE. Textual quotations should be avoided.

Neural Networks in Accounting and Finance Research

June 26, 2006

Abstract

This study is an introduction to the modern statistical modelling tools known as Neural Networks, and to their applications in accounting and finance research. It emphasizes the description of two kinds of networks: The Multi-Layer Perceptron (MLP) and the Kohonen's Map of Patterns. These are the most used Neural Networks nowadays. They also represent the two main views or branches of Connectionism. In this study we omit references to binary-threshold networks and those intended solely to study brain functions.

So far, expectations about Neural Networks were related to the modelling of difficult relationships (pattern recognition) or the mimicking of the brain. The reason for using Neural Networks in Finance and Accounting research is not just the need of simple and versatile, yet powerful, tools able to cope with the complexity of some relationships. It is also the fact that some Neural Networks exhibit unique characteristics potentially valuable. For example, some internal representations of data often are meaningful and an important way of acquiring knowledge from past experience.

Neural Networks in Accounting and Finance Research

This study is an introduction to the modern statistical modelling tools known as Neural Networks, but only to the extent of their usefulness in accounting and finance research. There are already many articles and books available on the subject. We limit this review to the kind of Neural Networks interesting for our study. Therefore, we omit references to binary-threshold networks and to those intended to study brain functions.

This presentation emphasizes the description of two kinds of networks: The Multi-Layer Perceptron (MLP) and the Kohonen's Map of Patterns. Incidentally, these are also the most used Neural Networks nowadays. They also represent the two main views or branches of Connectionism.

So far, expectations about Neural Networks are related to the modelling of difficult relationships (pattern recognition) or the mimicking of the brain. The reason for using Neural Networks in Finance and Accounting research is not just the need of simple and versatile yet powerful tools able to cope with the complexity of some relationships. It is also the fact that some Neural Networks exhibit unique characteristics potentially valuable. For example, their internal representations of data often are meaningful and an important way of acquiring knowledge from past experience.

Traditional tools for knowledge acquisition seem not to fit well in many financial applications. The nature of accounting and financial relations, where the input variables are continuous-valued and stochastic, makes it difficult for the usual expert systems based on symbolic computation to deal with. Observations such as those found in stock returns, or data organized in accounting reports, cannot be efficiently used by actual expert systems as a source of knowledge. Neural Networks can provide self-explanatory results, along with improvements in performance.

Contents: Section 1 contains an historical note written so as to give the necessary perspective of the development of these tools. Section 2 is an introduction to Neural Networks. Section 5 explains where Neural Networks are seemingly interesting in Finance and Accounting research.

1 Historical Notes

This section is dedicated to explaining the genesis of Neural Networks and the role they play in knowledge acquisition. Only an understanding of their origins and prospects can lead to the formation of an opinion about the interest of these tools in Accountancy and Finance.

The history of Neural Networks is not a common one. Known as such from the early forties, they trod an adventurous path with periods of intense enthusiasm and almost total eclipse. The actual developments, which began in 1985, are partly the outcome of a much expected discovery.

Learning theory and the learning tools known as Neural Networks are the result of two lines of research which began their paths very early — about the Second World War — and remained closely associated until the decade of the sixties. The first of such lines was technical. It included mathematicians and engineers trying to build what is known as the Optimum Filter. They used concepts extracted from Tele-communications: Linear Systems, Stochastic Optimization and Information Theory. The second line was speculative and its goal was the building of artificial machines similar to the human brain. This science was known, and still is, as Connectionism. In the next paragraphs we follow the development of both.

The Optimum Filter and other automatic learning devices: A filter is a tool able to separate, in a continuous flow of information, the real signal from the randomness attached to it. The study of filters is typical of the Information and Communication sciences. However, the problem of filtering is very general. Optimum filters are similar to automatic controllers or predictors. The problem of building them is similar to the problem of building a general statistical modelling tool able to separate pure randomness from real features of the data without the intervention of experts. For example, what engineers call a linear filter could be described as a linear regression in a time-series context. Past observations are used to predict future events.

Filters are optimal according to a criterion, in the same sense a regression minimizes the squared error. But they often include other criteria, like stability, as well. A learning or adaptive filter is the one which adapts its characteristics in accordance to changing inputs, so as to obey one or more criteria. In the linear case it would be a sort of adaptive regression able to change the slope and the intercept when the data changes.

Learning filters accomplish behaviour modifications without external intervention in its operation. The number of parameters engaged in the modelling and even the amount of non-linearity introduced, are selected just by the influence of the input and responding to its requirements. An automatic learning system acts like a dedicated controller whose experience of the underlying structure of the process improves as the process unfolds. Additional information concerning its structure and features causes the controller to adapt himself to the process's behaviour.

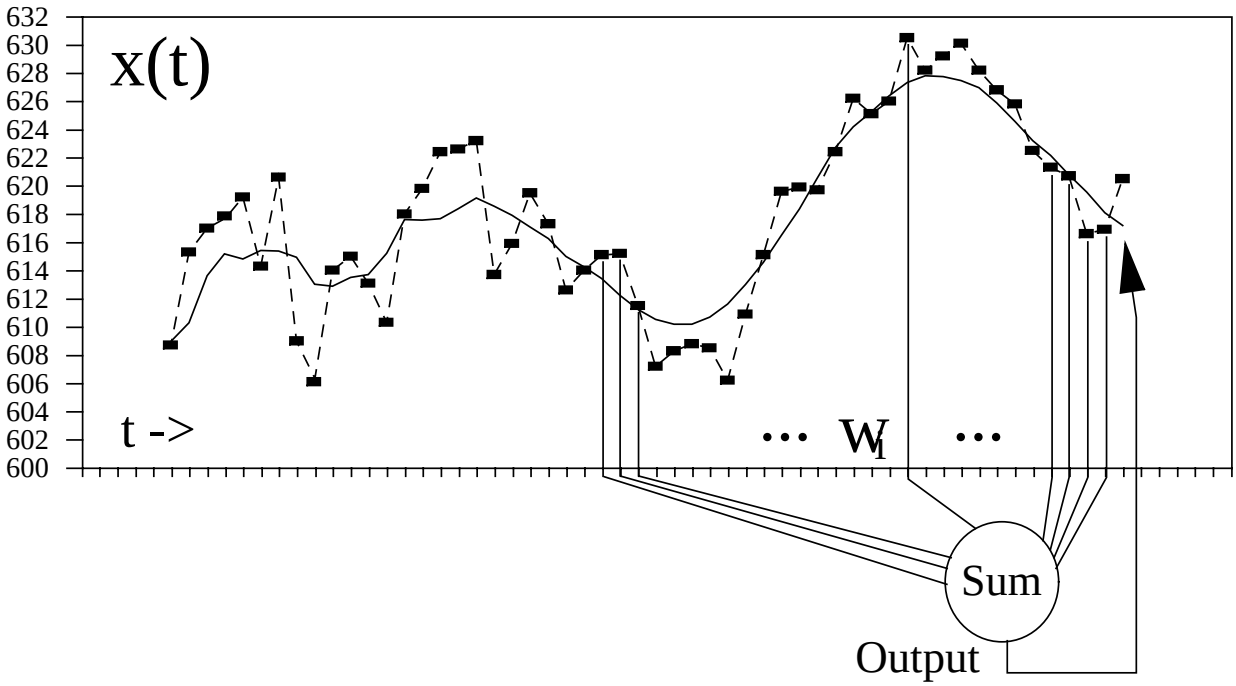

Figure 1: Schematic representation of the simplest Wiener filter. An inner product of adjustable parameters W with the past history of a time process X is used here to approach the actual event x_t. The solid line is the output of the filter.

Such tools are valuable in many areas but especially in communications and in control. They had their origin as solutions to problems posed by military automata. Nowadays they are also considered as potentially interesting for knowledge acquisition and machine learning. In fact, it is the very nature of the process which eventually emerges and becomes transparent when described by the set of parameters used for modelling it, along with the *a-priori* assumptions used. The amount of knowledge provided in this way often is more interesting than the model itself.

Early studies on Learning: In October 1941, Bell Laboratories and the Massachusets Institute of Technology (US) engaged Norbert Wiener and other researchers in an intense effort to design automatic devices which could track a plane or a ship, compute the main features of its trajectory and predict where it would be by the time the shell or bomb had travelled to the target area. The conceptual basis of this research became the origin of the early automatic learning systems.

Norbert Wiener had at that date a large experience on building filters. During the 30's he idealized, along with Y. W. Lee, a network of circuits able to perform convolutions of incoming signals. For a signal $x(t)$, t being a discrete time counter, such networks would

produce an output $O(t)$

$$O(t) = \sum_{i=1}^{M} x_{t-i} \times w_i \quad , w_i \text{ being adjustable parameters.}$$

M is known as the delay-line size. It controls the amount of memory about the past history of the signal that is incorporated into the filtering (see figure 1).

With such devices Wiener and Lee were able to perform many interesting tasks like the solving of partial difference equations. They were also able to design a linear filter of any frequency response just by modifying the parameters w_i.

For the prediction of plane trajectories Wiener used an improved version of the same basic networks attached to a mechanism of feedback. The position of a target was computed by the net and compared with its real position. From here a measure of error was obtained. Then, the parameters w_i were updated so as to minimize such error. This procedure was carried out interactively. The final result was that the trajectory of the target would be learned by the set of w_i.

Such a simple mechanism, which in practice didn't succeed, nevertheless became the basis of modern filters and the paradigm of automatic learning for more than twenty years. Some of the modern Neural Network learning rules are also based on it.

Further developments: In order to identify or to recognize the pattern for automatic learning, it is necessary to build a mathematical model of the process to be learned. Kolmogoroff (1942) [15] and Wiener (1949) [48], assumed first that the process was linear. Then, they demonstrated that the filtering and the prediction of stochastic series were special cases of the same learning problem. Their work could be described as the finding of a general recipe for the building of learning systems. The significance of such work is stressed by Y. W. Lee (1964) [18]:

> Wiener's theory of optimum linear systems is a milestone in the development of communication theory. The problems of filtering, prediction and other similar operations were given a unity in formulation by the introduction of the idea that they all have in common an input and a desired output. Then, the minimization of a measure of error, which is absent in classical theory, was carried out. The entire theory, from its inception to the final expressions for the system function and the minimum mean-square error is invaluable in the understanding of many problems in a new light.

As originally formulated, Wiener's early methods are applicable only to linear time-invariant dynamic processes which are to be optimized by a Least-Squares criterion. Booton (1952) [3]

extended Wiener's work to the optimization of linear time-varying dynamic processes possessing either time-invariant or time-varying statistics. Kasakov (1956) [12], Shen (1957) [36], and others, treated nonlinear feedback control systems with random inputs using stochastic learning techniques.

Wiener's latest work on this subject, *Non-Linear Problems in Random Theory* (1958) [49] opened up the path for a theoretical approach to self-organizing and adaptive systems. In his framework, the complexity of the system's repertoire of available non-linearity increases as the learning process develops so as to maximize the flow of new information about the structure of the process thus creating an internal model of it. In a restricted sense, if some input and output functions represent the behaviour of an unknown process, the Wiener system will organize itself into a model of this unknown mechanism, provided statistical regularities exist in the process. The basic tool for such organization is, of course, the ability to abstract those regularities from the stochastic series on which it is to operate.

Limitations of analytical learning systems: Being analytical, the Wiener solution cannot avoid some lack of generality. Assumptions must be made about the statistical nature of the input. If not, it would be impossible to apportion analytically the parameters between input and output. Only Gaussian processes and a few more classes of random processes are correctly modelled by this method. And the modelling of the signals is made — in the later versions — using Volterra functionals, that is, Taylor series with some limited amount of memory of past events (see Volterra 1930) [45].

When a system becomes optimal only given the restriction that it must belong to a specified class, the kind of information that such a system can identify and use is also restricted. A linear system, for instance, can produce a significant improvement in mean-square error reduction, only if the spectral densities of the signal and the randomness attached to it are different, since it cannot use any other information. Therefore, an optimum linear system, in the Wiener's sense, is no better than an optimum attenuator. Higher order modelling requires non-linear systems because dissimilarities in the characteristics of stochastic series, which the linear system would ignore, can now be used to reduce mean-square errors.

Although no analytic general solution exists for the general learning model some broad cases have been explored. In the U.K. Denis Gabor built in 1960 his *Universal Non-Linear Filter, Predictor and Simulator which Optimizes Itself by a Learning Process* [6]. It was an application of Wiener's later work. In the US, Shen and Rosenberg (1964) [35] used the same principles.

Many military and tele-communication applications of these early attempts followed.

They were analytical-based dedicated automata to be used whenever computational speed was required. These "Wiener-Volterra Systems" are not very flexible nor very good in generalisation for, after all, they use polynomials to approach the data. But being analytical, they avoid problems of convergence and are easily built into very fast hardware. A good review of such systems can be found in Schetzen's book (1980) [32].

Neural Networks seek the same goal. But they are not based on analytical optimization. Stochastic search techniques are used instead to discover, in the parametric space, a point obeying a desired condition. They generalise better but their learning process is much slower.

Brain Mimics: It is bizarre that a practical application of Wiener's aspirations was at length provided, not by any mathematical analysis, but by a few neurophysiologists trying to build models of some brain functions like reasoning and recognition.

Artificial neural models emerged forty years ago as a broad mimic of the real neural structure of the brain. The paradigm of Connectionism's early work is the Hebb's Rule. Strongly influenced by Behaviourism and other theories accepted at that time, Donald Hebb wrote in 1948 a book, *The Organization of Behaviour* [8], proposing a plausible mechanism by which learning could take place in the brain. The Hebb's Rule simply states that whenever two neurons are excited at the same time the connection between them strengthens. Many Neural Network learning rules had their origin in this mechanism or in variations of it. However, the strong theoretical basis of connectionism has been provided by mathematicians rather than by Psychologists.

Like Wiener, Dr. Warren McCulloch was a mathematician interested in practical problems. He first met Wiener in 1942 and their collaboration lasted for a few years. McCulloch was mainly interested in the organization of the cortex of the brain. Working with him was Walter Pitts, a student of logic and biophysics. In 1943 Pitts moved to the MIT for reading with Wiener. They worked together on pattern recognition until 1948.

Wiener's ideas on filtering and automatic learning must have inspired the first paper that McCulloch and Pitts published. It was *A logical Calculus of Ideas Immanent in Nervous Activity* (1943) [21], formalizing the intrinsic structure of the neural process leaving aside its biochemistry. The calculus was very revealing. Using only concepts like the firing of neurons, excitatory and inhibitory connections, synaptic delays, all-or-none processes, it was possible to show that the specifically biologic aspects of the nervous system are irrelevant to the understanding of perception. But the most important aspect of this work is the parallelism it establishes between the Turing Machine and the brain.

The concepts of Turing's Machine is significant, not only from the purely

> analytical standpoint of mathematical logic, but even from the standpoint of the neurophysiological understanding of the human mind (McCulloch 1965 [22] page 35).

A consequence is that the mind can compute all and only those numbers a Turing Machine does. After this, McCulloch and Pitts turned their attention to the problem of the recognition of patterns. Wiener describes it as an interrogation:

> What is the mechanism by which we recognize a square as a square, irrespective of its position, its size, its orientation? (Wiener 1961 [50] page 18).

In 1947 they offered a theoretical description of the neurophysiological mechanisms for pattern recognition in the brain, in their paper *How We Know Universals* [27]. They suggested mechanisms similar to those of modern Neural Networks for explaining the ability of the brain to recognize. This paper strongly influenced Wiener's thoughts on Learning and Control, being an important point of view for the work he was about to undertake.

The second generation of connectionists: In the 50's the dominant research in Neural Networks was led by Frank Rosenblatt at Cornell, US Based on the theoretical developments of McCulloch, he built a network called *Perceptron*, supposed to be able to learn to recognize physical objects by looking at them (see Rosenblatt 1961 [30]).

In 1959 Bernard Widrow [47] developed at Stanford, US, an adaptive linear filter called Adeline based on neuron-like elements. The Adeline and its more sophisticated successors were used for a variety of applications including the recognition of speech and characters, weather forecasting and adaptive filtering. The Adeline was also the first Neural Network to be used in a practical real-world application, the automatic elimination of echoes in phone lines. With Widrow, the two branches described above made a mutual recognition: Engineers became interested in Neural Networks.

In the sixties, the results of using Neural Networks were promising enough to attract general attention. Particularly, Rosenblatt's Perceptron generated real enthusiasm in the scientific community.

The emergence of Symbolic Computation: Such an enthusiasm lasted for a short period. In 1969 Minsky and Papert published *Perceptrons* [24] showing that Rosenblatt's two-layer Perceptron, being linear, would not recognize patterns involving higher order effects like those which occur in the logical "exclusive-OR" problem (see figure 2). In order to correctly classify patterns separated by non-linear boundaries, Minsky and Papert showed

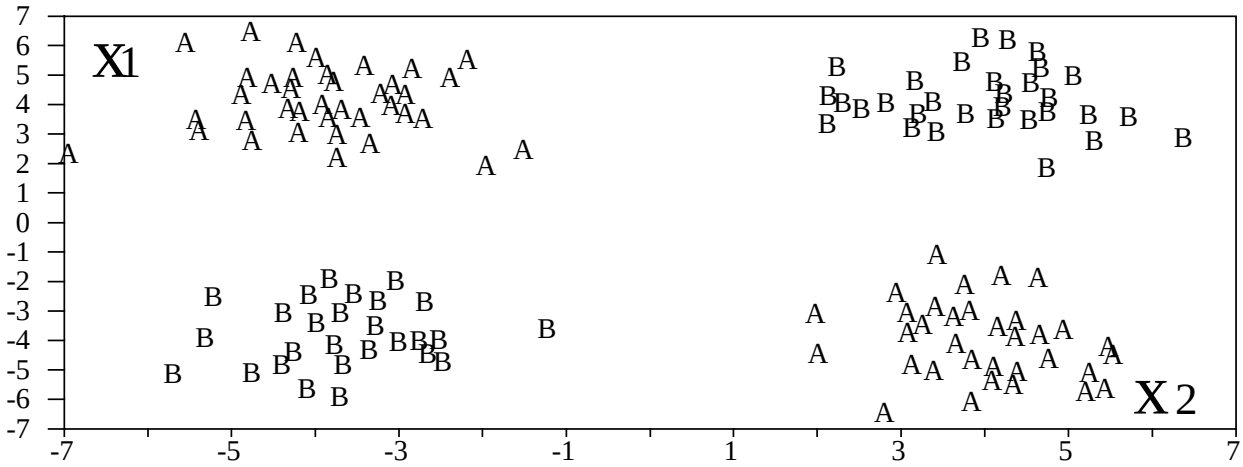

Figure 2: The statistical version of the logical exclusive–OR problem used by Minsky and Papert to discard the early Perceptron. No linear frontier in the space of x_1 and x_2 can separate the two groups A and B in spite of their clear separability.

that Perceptrons would need more than two layers of neurons and the introduction of non-linear transfer functions. At that time no one knew how to build a general learning rule able to adapt the connections between internal neurons.

After that, interest in neuro-models languished. The attention of the research concerned with Learning was directed towards the emerging tools provided by Artificial Intelligence. During twenty years, Connectionism was confined to a few laboratories, mainly concerned with the brain itself: James Anderson, at Brown University, US, revived the Hebbian principle in his *Linear Associator* (see Anderson 1970 [2]). Teivo Kohonen, in Helsinky University, Finland, also envisaged a modified Hebbian principle known as *Competitive Learning*, for creating self-organizing maps of patterns (see Kohonen 1974 [13]).

During the seventies and early eighties the dominance of symbolic methods was overwhelming. Machine Learning and Knowledge Acquisition became synonymous with Symbolic Computation. The emergence of the Computer Sciences as an independent branch, the fast progress in the speed of conventional machines, the existence of generous funding, all this turned the attention of the scientific community towards tools based on Discrete Mathematics and Logic. Research programs based on analogue tools were discontinued and most of the earlier contributions for Machine learning were forgotten. Machine Learning went exclusively symbolic.

The result was a delay in the course of the development of these subjects. And at length, a muddling of concepts and techniques. For example, some researchers tried seriously to use discrete, hierarchical, learning methods like Quinlan's ID3 (see Quinlan 1979) [28]) as an alternative to simple linear regressions in straightforward problems involving the prediction of

continuous-valued variables ([23], [29]). These authors clearly put the finding of hierarchical rules ahead of any other consideration, as no other choice was available for Machine Learning.

Another result of these years was the narrowing of views and goals inside the small connectionist community. Connectionists became strictly concerned with the mimic of brain functions as a goal, frequently denying the idea that such mimic could be used also as a source of inspiration for the building of useful learning algorithms. A typical example is Stephen Grossberg. He devised an *Adaptive Resonance Theory* (Grossberg 1987 [7]) leading to self-organized memories with local characteristics. It is a plausible mechanism for the brain with small practical applicability.

Even nowadays, the followers of Connectionism will discard any model for learning if it is not plausible enough as a replica of the brain. One of the most demolishing things anyone can say about a new Neural Network is that it is not enough brain-like. As a consequence, many recent algorithms are local, self-organized and altogether with little interest for this study.

The return of Neural Networks: The MIT was one of the few places were the interest in analogue learning never vanished. In 1978 John Hopfield initiated a collaboration with its Centre for Biologic Information Processing. As a result, he presented in 1982 a paper to the Academy of Sciences of the US about a new neural model. It was the first paper on connectionism accepted in this body since the 60's.

Hopfield's model [11] introduced a conceptual basis for neural learning in terms of energy. It also established a parallelism with Ising models (Spin Glass Physics). Hopfield's net uses fully interconnected neurons that seek a minimum of energy. A few years later, Geoffrey Hinton in Toronto and Terrence Sejnowski in the John Hopkins University, US, developed a modified version of the Hopfield net they named *The Boltzmann Machine* (1983) [10], able to escape from local minima during learning and with the remarkable quality of being trainable even when having hidden layers. Hopfield's net and the Boltzmann Machine considerably revived the interest of the scientific community on Neural models.

The breakthrough came in 1985 when David Rumelhart, a professor of psychology at Stanford, US, and James McClelland, a psychologist at Carnegie-Mellon, along with other members of *The Parallel Distributed Processing Group* (known as PDP) devised a learning scheme that would allow multi-layered Perceptrons to feed back deviations from correct response to more than one layer of neurons. Their scheme became known as *The Back-Propagation Algorithm* [31]. It allows the training of all nodes inside a Multi-Layer Perceptron (MLP), even the internal ones.

Much interesting research followed. For example, the link between Back-Propagation and the continuous-valued version of the Hopfield paradigm was established shortly afterwards by Luis B. Almeida at INESC, Lisbon (1987) [1]. Almeida generalised Back-Propagation so as to make possible the learning in networks with any topology. The original MLP were limited by a feed-forward topology in which the information flows only in one direction — from the input to the output layer. This *Recurrent* back-Propagation is especially adequate for tasks requiring some amount of memory of past events like systems identification, the reconstruction of missing cases and the simulation of dynamic systems.

As predicted by the early research, the MLP turned out to be a very powerful and versatile modelling tool, able to solve Minsky and Papert's exclusive-OR problem and many other complicated ones. The MLP is, in practice, a general learning tool as Wiener foresaw it. This fact, interesting as it is, induced a sudden and somehow non-proportional enthusiasm and renewed the interest in models based on Connectionism.

Indeed, one of the reasons for such a renewal of interest in Neural Networks was the realization that Symbolic Computation, when tackling even the simplest problems involving pattern recognition, was severely limited. The task of recognizing "a square as a square irrespective of its position" turned out to be very hard for tools based on logic, rules, hierarchical structures or other concepts of Artificial Intelligence. Typical problems of this kind, like the recognition of voice and handwritten information, received a second chance of improving by using Neural Network techniques instead.

Neural Networks today: Nowadays, Neural Networks are being used for finding solutions for difficult problems of recognition of voice and image, in military applications, in Biology and Chemistry, in Medicine and many other fields. Sejnowski's *NETtalk System* (1986) [33] is often mentioned as a reference point for assessing what Neural Networks can do in these fields. This program converts text to speech and, connected to a speech synthesiser, it pronounces typewritten words. It learns from examples of text together with its spoken form. Over a few hours of such training, it progresses from a formless babble to intelligible English.

It is clear that, again, there is no correspondence between the real possibilities offered by Neural Networks and some extravagant expectations about them. The same strong motivations which led to non-realistic views about Artificial Intelligence during the last decades are now working in the direction of Neural Network research:

> The reason why the US AI community (academic as well as commercial) has taken up the neural-net model so enthusiastically is quite straightforward. It is

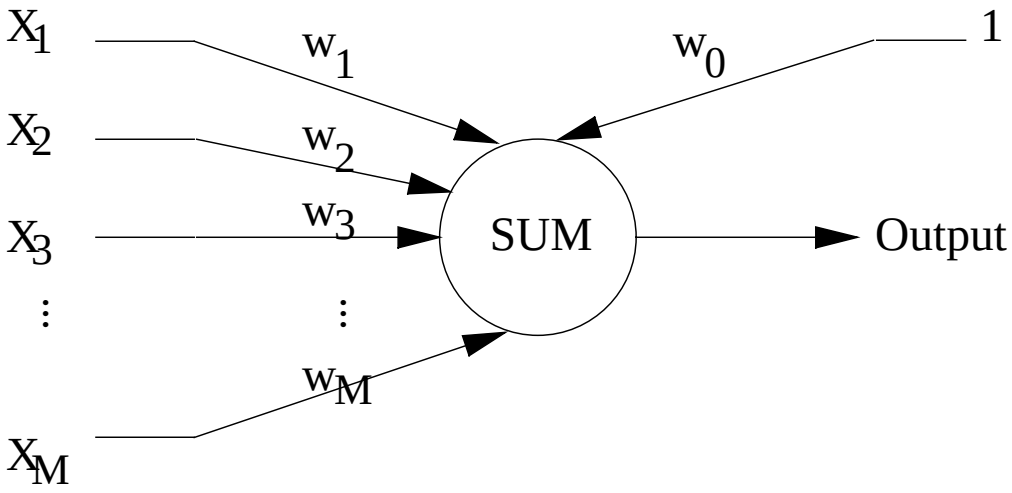

Figure 3: A neuron or node, the basic element of Neural Networks. It implements an inner product,$X \cdot W$, of an input vector, X, with another one of adjustable parameters W.

> primarily because the Department of Defense has decided that neural-net computing is a high-priority strategic technology. As an example, the UCLA (University of California, Los Angeles) AI lab has recently started ten new projects concerned with neural networks while seven symbolic AI projects are due to be terminated shortly. This switch did not come from inside the university. It happened as a result of strong prompting from DARPA and other funding bodies (Forsyth 1988 [5], page 12).

Thus, when assessing the possibilities offered by Neural Networks it seems important to have a clear idea about their applicability, strong and weak points and clear shortcomings. The most quoted of these shortcomings is the apparent non-ability of Neural networks to produce interpretable and exportable models like those based on rules and structures.

In this study we comment on several examples of Neural Networks in accounting and finance research. In some of them, the resulting model is simple and its structure is interpretable. Such an interpretability makes it attractive. But it is interpretable because it is simple. In many others, the obtained model is complex and cannot be decoded. But this is not a shortcoming since the model is important by itself, not because of its interpretability. Knowledge doesn't have to be interpretable in all cases. The most complicated pieces of knowledge couldn't possibly be translated into simple structures.

2 The Structure of Neural Networks

"Neural Network" is the name of a vast number of different heuristics, having in common the fact that all of them are aimed at learning from past experience, and their topology is inspired by the way the brain is supposed to work. If a sample containing input and related outcome variables represent an unknown relationship, a Neural Network will model it by successive approximations. Such process is known as learning or training.

Topologically, Neural Networks are lattice structures of simple computational elements called *neurons* or nodes. The connections between nodes (the *weights*) can be strengthen or weakened during the training process, by means of simple, local, rules, causing the network, as a whole, to become more and more similar to the relationship present in the data.

A typical node sums M weighted inputs. The result is then used as input for several other nodes. That is, input variables labeled $x_1, \cdots, x_M$ are applied through a set of associated weights, $w_1, \cdots, w_M$, to a node. The weighted sum of the inputs, $s = \sum_{i=1}^{M} w_i \times x_i$, is the output of the node. This output is then used as an input for several other nodes, and so on. Figure 3 on page 11 shows the simplest node.

Nodes are often arranged in *layers*. In a layer of nodes, the M inputs are connected to the N nodes, that is, a weight w_{ij} exists, linking each input $x_i, i = 1, M$ with each node $j, j = 1, N$. In a layer, outputs of nodes are $s_j = \sum_{i=1}^{M} w_{ij} \times x_i$. Layers can be combined into a network (figure 4), by cascading two or more layers of nodes so that the outputs of one layer become inputs to the next one. These simple Neural Networks are similar to those built by Widrow in the sixties [47]. They are limited in their applications because no non-linearity is introduced during training. They learn only linear relationships. Modern networks use non-linear functions associated with each node.

Neural Network classification: Neural Networks can be completely specified by three characteristics:

- The topology of the net: The number and position of the nodes and the way connections between nodes are allowed.
- The node's transfer function: The non-linear operation that each node performs on its output before delivering it to other nodes.
- Learning rules: The particular algorithms used for training the network through a steady adaptation of its weights.

According to their topology, Neural Networks often belong to one of these families:

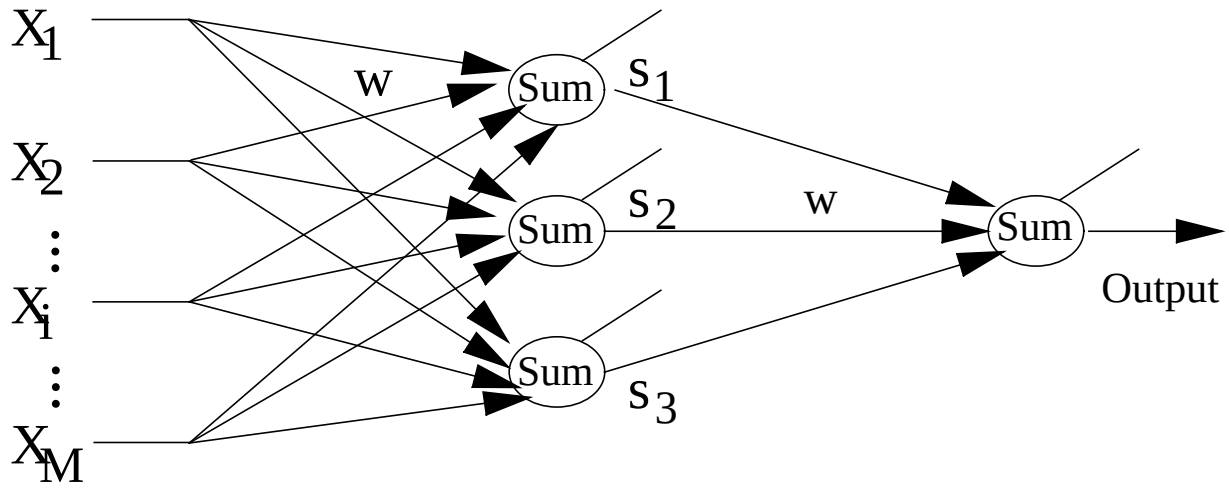

Figure 4: A simple Neural Network containing the input, a hidden layer of nodes and one output node.

- Only feed-forward connections: The nodes are organized in layers and each node in one layer connects only to next layer's nodes. The Perceptron belongs to this group.
- Intensive wiring: Each node's output is connected with all the other nodes's inputs. An example is the Hopfield net.

According to the transfer functions, there are two main kinds of Neural Networks: Continuous non-linearity, for analog processing; and hard-limiter threshold, for digital processing. The Multi-Layer Perceptron uses continuous non-linearity. The early Hopfield and Perceptron networks used hard-limiter thresholds.

Finally, according to their training rules, Neural Networks can be

- Supervised — There is a learning process in its strict sense, as in the Perceptron and Hopfield networks: Each learning case consists of a vector of inputs and a corresponding vector of desired outcomes. The Neural Network learns the relationship, if any, between inputs and outcomes, as present in the data.
- Unsupervised or self-organized, as in the Kohonen Map of Patterns. The learning process does not require explicit supervision: The network organizes itself according only to the inputs, becoming a map of their density function.

The Multi-Layer Perceptron is, by far, the most popular Neural Network. We shall devote some attention to it now.

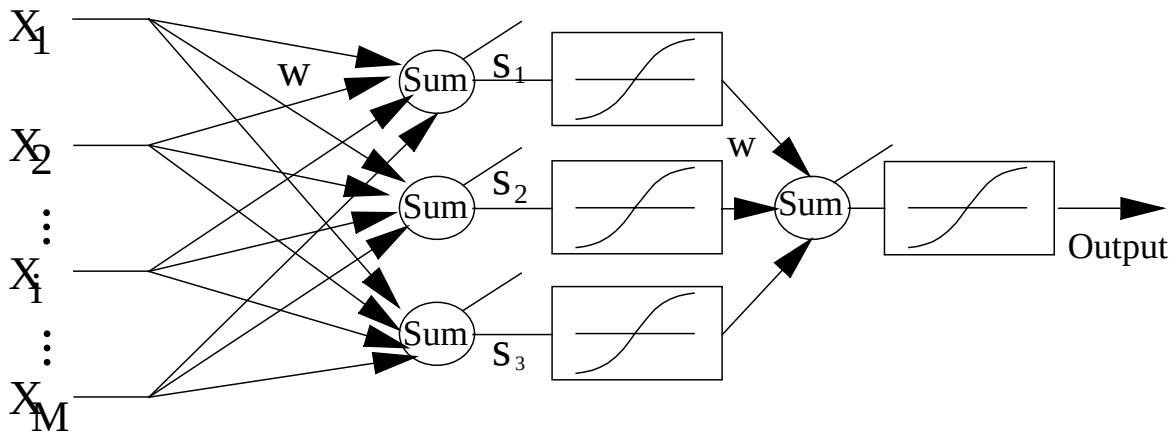

Figure 5: The Multi-Layer, feed-forward Neural Network known as the MLP. In this case there are inputs, one hidden layer, and one unique output node.

3 The Multi-Layer Perceptron

The Multi-Layer Perceptron, widely known as the MLP, is a supervised learning Neural Network. Topologically it is a layered feed-forward configuration: Nodes are arranged in layers and each node's output is connected to next layer's inputs. No intra-layer connections exist, nor any feedback paths from an output to earlier layers. Figure 5 on page 14 represents a three-layer Perceptron with only one output node.

The back-propagation of errors by means of an iterative gradient-descend algorithm allows training by minimization of the mean-squares differences encountered between the actual and desired output.

In the Multi-Layer Perceptron the output of the, j^{th}, node is not just a simple weighted sum of inputs. After the summation $s_j = \sum w_{ij} \times x_i$ the result is submitted to some continuous differentiable non-linearity $f(s_j)$ becoming $o_j = f(\sum w_{ij} \times x_i)$. It is common to use sigmoid-like functions as $f(s)$. In this case the output of the j^{th} node would be:

$$o_j = \frac{1}{1 + \exp\left[\frac{-(s_j + \theta_j)}{\theta_0}\right]}$$

Figure 6 on page 18 displays one of such functions.

The parameter θ_j acts as a threshold or bias. The effect of a positive θ_j is to shift the sigmoid to the left along the horizontal axis. θ_0 modifies the steepness of the slope in the sigmoid. Large θ_0 produce smooth thresholds. Conversely, small θ_0 produce sharp thresholds which can be almost like the hard-limiter ones. These non-linearities are known as transfer or activation functions. In the same line, $1/\theta_0$ is known as the gain.

Mean-Squares error: We now consider the last layer of nodes as the k^{th}. The layer immediately before this one is the j^{th} and the one before that is the i^{th}.

When learning, the net is shown a vector of inputs, X_p, from the p^{th} example. Then, the algorithm adjusts the set of weights and the thresholds in each node in a way that makes the effective output of the net, o_{pk}, as close to the desired outcome, t_p, as possible. Once this small adjustment has been accomplished we show another input vector and the corresponding outcome and ask that the net learns this new association as well. At length, we are making the net find a single set of weights and biases that will satisfy all the input-outcome pairs present in our learning set.

In general, the final resulting outputs o_{pk} will not succeed in approaching t_p exactly. For the p^{th} example, the squared error is

$$E_p = \frac{1}{2}\sum_k (t_p - o_{pk})^2 \tag{1}$$

and the average total error will be

$$E = \frac{1}{2 \times N}\sum_p \sum_k (t_p - o_{pk})^2 \tag{2}$$

in which the $1/2$ scale is introduced for facilitating the algebra at a later stage and N is the number of examples in the learning set.

3.1 The Delta Rule: The Last Layer's Nodes

The Delta Rule is a stochastic version of the steepest-descent iterative optimization algorithm. It has been used in the early Perceptrons. Strictly, it applies only to the learning process taking place in the last layer of the MLP. The learning taking place in layers other than the last one, is accomplished using a generalised version of the Delta Rule. We shall introduce the Delta Rule firstly, by applying it to nodes in the k^{th} layer of the MLP, the last one. Next, it will be easier to explain its generalized version.

Starting with an arbitrary set of W values, every example in the learning set will be considered in a random order and its output calculated. Then, the difference between this output and the corresponding outcome will be used to correct every parameter in W by a small amount. The procedure will be repeated for all examples in the learning set again and again, until a minimum square difference exists between outputs and outcomes. In general, different results emerge depending on whether the gradient search is carried out on the basis of E_p or E. A true gradient search should be based on minimizing (2). In practice, that is seldom the procedure adapted.

Convergence is achieved by changing the values of W so as to diminish the error. This is carried out by taking incremental changes Δw_{kj} proportional to $-\partial E/\partial w_{kj}$. That is, given a small η,

$$\Delta w_{kj} = -\eta \times \frac{\partial E}{\partial w_{kj}}$$

Since the error E can be expressed in terms of the outputs o_k and these outputs are a non-linear function $f(s_k)$, we can use the chain rule to evaluate the above partial derivative:

$$\frac{\partial E}{\partial w_{kj}} = \frac{\partial E}{\partial s_k} \times \frac{\partial s_k}{\partial w_{kj}}$$

Notice that

$$\frac{\partial s_k}{\partial w_{kj}} = \frac{\partial}{\partial w_{kj}} \sum_j w_{kj} \times o_j$$

We now define

$$\delta_k = -\frac{\partial E}{\partial s_k}$$

as the rate of change of the error with respect to s_k. So, we can write

$$\Delta w_{kj} = \eta \times \delta_k \times o_j$$

This expression contains the rule for updating the weights linked with the last layer of nodes. It is known as the Delta Rule. η is an arbitrary increment. It is a small value selected so as to ensure a smooth, yet fast, learning.

To compute $\delta_k = -\partial E/\partial s_k$ we use again the chain rule and obtain two terms:

$$\delta_k = -\frac{\partial E}{\partial s_k} = -\frac{\partial E}{\partial o_k} \times \frac{\partial o_k}{\partial s_k}$$

The first term expresses the rate of change of the error with respect to the output o_k and the second one expresses the rate of change of the output of any node in the last layer with respect to its input.

These two terms are easily obtained. From (1),

$$\frac{\partial E}{\partial o_k} = -(t_k - o_k)$$

The second partial derivative depends solely on the transfer function we are using:

$$\frac{\partial o_k}{\partial s_k} = f'(s_k)$$

Hence, whenever we accept a Least-Squares success criterion, the deviations $t_k - o_k$ from the desired outcome can be viewed as the rate of change of the error with respect to the node's output. Notice that when the criterion is a different one this relation has to be re-written.

For any node in the last layer we can write

$$\delta_k = (t_k - o_k) \times f'(s_k) \tag{3}$$

hence

$$\Delta w_{kj} = \eta \times (t_k - o_k) \times f'(s_k) \times o_j \tag{4}$$

The original Perceptron of Rosenblatt, having only one layer of nodes, would learn to optimize an internal model of the data with steady improvements, Δw_{kj}, of the weights w_{kj} by means of the described algorithm. Adaptive filters and some versions of the Adeline also use this learning scheme.

3.2 The Generalised Delta Rule

The general solution of the non-linear modelling problem came with the use of internal layers of nodes acting as intermediate maps able to apportion as much piece-wise non-linearity as needed for modelling the relation. In the modern version of the Perceptron, the MLP, several layers of nodes are matched in cascade so that the output of a previous one is the input for the next. Theoretically, such a device is able to form any continuous smooth map if enough number of internal nodes are provided.

Of course, in this new situation the problem is to discover a suitable way for optimal parameter finding. The Delta Rule described above cannot be used in the learning of relations by more than one layer of nodes. As mentioned, the solution of this problem has been provided by neuro-biologists [31]. The method is a generalisation of the Delta Rule already introduced, known as Back-Propagation.

When weights are not directly linked to the output nodes we still write

$$\Delta w_{kj} = -\eta \times \frac{\partial E}{\partial w_{ji}}$$

and proceeding with the same formalism, based on the chain rule, we have, as before:

$$\begin{aligned}
\Delta w_{kj} &= -\eta \times \frac{\partial E}{\partial s_j} \times \frac{\partial s_j}{\partial w_{ji}} \\
&= -\eta \times \frac{\partial E}{\partial s_j} \times o_i \\
&= \eta \times \left(-\frac{\partial E}{\partial o_j} \times \frac{\partial o_j}{\partial s_j} \right) \\
&= \eta \times \left(-\frac{\partial E}{\partial o_j} \right) \times f'(s_j) \times o_j \\
&= \eta \times \delta_j \times o_i
\end{aligned}$$

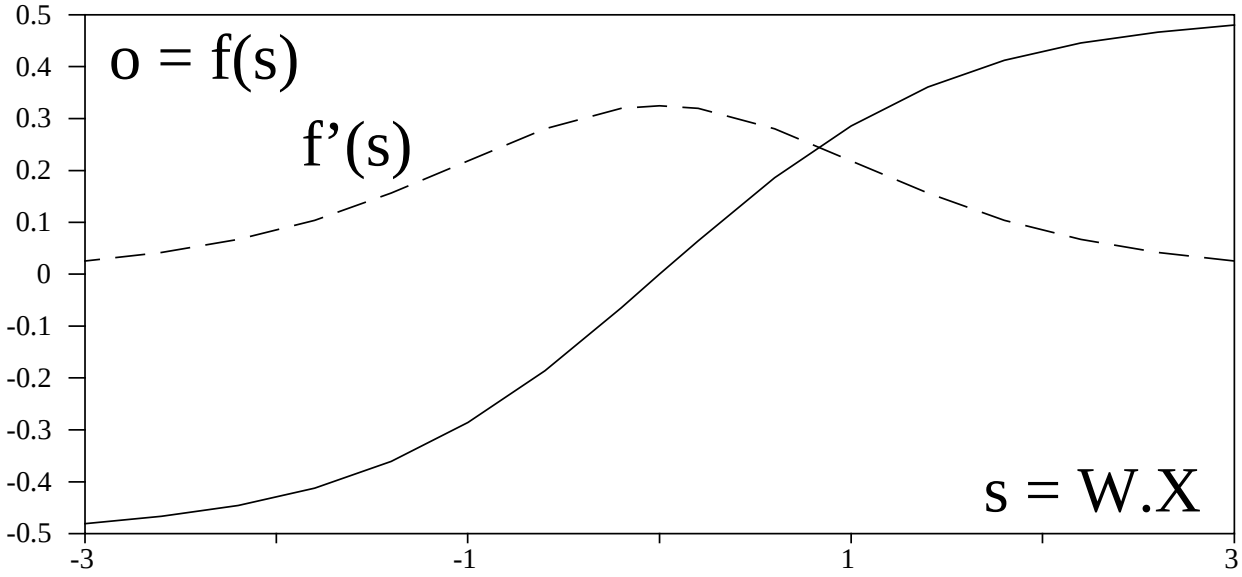

Figure 6: A sigmoid-like function can be used as activation or transfer function in the nodes of an MLP. $f(s)$ is the solid line and its derivative, $f'(s)$, is the dashed line. In this case, f spans the interval $\{-0.5, +0.5\}$.

an expression similar to equation (4). In the case of hidden nodes we cannot evaluate $\partial E/\partial o_j$ directly. However, we can write it in terms of known values obtained from the nearest layer:

$$\begin{aligned} -\frac{\partial E}{\partial o_j} &= -\sum_k \frac{\partial E}{\partial s_k} \times \frac{\partial s_k}{\partial o_j} \\ &= \sum_k \left(-\frac{\partial E}{\partial s_k} \right) \times \frac{\partial}{\partial o_j} \sum_m w_{km} \times o_m \\ &= \sum_k \left(-\frac{\partial E}{\partial s_k} \right) \times w_{kj} \\ &= \sum_k \delta_k \times w_{kj} \end{aligned}$$

Therefore we can write in the case of hidden nodes:

$$\delta_j = f'(s_j) \times \sum_k \delta_k \times w_{kj}$$

That is, the rates of change of the error with each node's s can be computed from the previous node's δ. Previous in the sense that they are closer to the output. The basic mechanism of Back-Propagation consists in making it possible to evaluate all the δ throughout the net just by beginning to evaluate them for the last layer and then proceeding backwards. The Back-Propagation algorithm evaluates firstly the δ_k using (3) and "propagates" these errors backwards along the net.

Notice that, when the $f(s)$ are sigmoids, hyperbolic tangents or similar logistic functions, then their derivatives, the $f'(s)$ assume a very simple formalism. For example, in the case of the sigmoid,

$$\text{since} \quad o_j = \frac{1}{1 + \exp\left[-\left(\sum_i w_{ji} \times o_i + \theta_j\right)\right]} \quad \text{then} \quad \frac{\partial o_j}{\partial s_j} = o_j \times (1 - o_j)$$

and the δ are given by these or similarly simple expressions:

$$\delta_k = (t_k - o_k) \times o_k \times (1 - o_k) \quad \text{and} \quad \delta_j = o_j \times (1 - o_j) \times \sum_k \delta_k \times w_{kj} \tag{5}$$

for nodes in the output layer and for nodes in the hidden layers respectively.

For implementing Back-Propagation, a step-by-step procedure is required. Firstly, an example is selected at random. Using it as input, the output of the MLP is calculated. Next, the Δw_{kj} are also calculated and the weights linking to the last layer are corrected. Using these corrected weights and the δ_k it is now possible to calculate the Δw_{ji}, that is, the correction to affect the weights linking to the layer before the last one. This scheme proceeds backwards until all the weights in the net received their first correction.

The whole procedure described above is now repeated for every example in the learning set and for as many "presentations" of the whole set as necessary to bring the overall error to a minimum. In successful learning, the net's error decreases with the number of presentations and it will finally converge to a stable set of weights.

Notice that the Back-Propagation is not a real gradient-descent algorithm since the weights are corrected by evaluating the errors E_p associated with each example (formula 1), not the overall error E of the sample (formula 2). It could be described as a stochastic gradient descent. In the case of correlated inputs, it is more effective than the classic algorithm. For small values of η the difference between the stochastic and the classic versions of the gradient-descent procedure vanish.

The increments η must be selected carefully. Too small η make the learning very slow and vulnerable to local minima. Too large η produce oscillations of the error during training. Recent versions of Back-Propagation don't use the same η for all the weights in a network. Instead, each weight has its own η, also adjusted during training according too the importance of weights to the diminishing of errors. This procedure makes training much faster and less prone to local minima. See Silva and Almeida (1990) [37] for details.

The most appropriate thresholds are learned by the algorithm just as any other parameter. The θ_j can be seen as the weight linking a constant input of 1 with the node. The use of hyperbolic tangents instead of the logistic form has no particular relevance except in the last layer. Sigmoids in the output layer constrain outputs to span the interval $\{0, 1\}$

while hyperbolic tangents constrain outputs to the interval $\{-1, +1\}$. But, of course, a simple manipulation can produce any desired interval. Output intervals must be selected in accordance with the outcomes.

The Generalised Delta Rule requires that the initial values of the weights are set to values different from one another. It is usual to set them to small random values. If the weights were all similar the net would be in a state known as "local minimum" and the learning wouldn't take place. Several runs of the training set will generally produce a minimization of E although nothing will prevent this heuristic from reaching other local minima instead of the overall solution (see Rumelhart's original paper for a discussion on this issue). In practice, the use of advanced techniques, like having one individual η for each weight, can greatly lighten the problem of local minima.

3.3 Valuable Characteristics of the MLP

In this section we summarize the characteristics of the MLP which seem valuable for knowledge acquisition and statistical modelling. This is an important issue since MLPs are very expensive in CPU time and attention from the operator. MLPs are the kind of tool which no one uses unless it is really necessary. Those characteristics are as follow:

Learning by stages: Many different algorithms are available for learning a relationship input-outcome from a set of examples. The MLP is different in that it approaches a relationship by stages, not directly. During the learning process, an MLP creates new sets of variables corresponding to different stages of the modelling of the desired relationship. A particular stage uses the variables from the previous one as input. Then, it makes an improvement towards the final modelling of the relationship. Finally, it outputs a new set of variables to be used as input for the next stage. The intermediate variables generated by an MLP are often referred to as *internal representations*.

The characteristic really specific to the MLP is this ability to model by stages, as it cannot be found in any other tool. However, the other desirable features of the MLP are not easily found, all of them, in the same tool. For example, some statistical algorithms perform stochastic non-linear optimization. But they have little control over the amount of non-linearity they use, or over the dimensionality allowed to model a desired relationship.

Robust Generalisation: The overall rules governing the generalisation ability in any statistical model also apply to the MLP. For instance, if the number of nodes (and therefore, connections) is large when compared with the variability to be modelled, the MLP behaves

just like a checking list (a storage device) or a saturated model. No generalisation can be expected. An opposite situation, very few nodes when compared with the variability to be modelled, would make the MLP recognize only broad families of features, without detail.

Using an internal layer with a variable number of nodes it is possible to control the basic information flow used in classification: Many nodes will produce great detail or even no generalisation at all; less nodes will improve generalisation. However, the MLP often exhibits a remarkable behaviour which greatly improves the generalisation beyond that expected for a given number of free parameters. This is caused by two factors: Nodes are prone to individuality, and the matching of the topology of the network with the structure of the relationship.

Any change in a weight's value is proportional to $\partial o / \partial s$. Hence, the changes are maximal for values of s impinging upon the central zone of the transfer function, the one having the largest slope. Figure 6 on page 18 illustrates this fact. It shows the shape of a sigmoid-like function and its derivative. Since changes in a weight's value are proportional to the magnitude of this derivative it turns out that mid-range values introduce large changes in the weights while extreme values make the net change little.

When the s are mostly in the mid-range of their transfer functions, the node under question is not yet trained or committed. It can turn up or down. Under these conditions the weights change rapidly. On the contrary, a committed node changes their weights little since the derivative is small.

The described feature is interesting since it shows that, inside a given topology and by the influence of a learning set, nodes tend to acquire a stable state and remain there. Hence, it is expected that each node on a fortunate model will capture important features of the relationship. The literature refers several of such cases. In terms of knowledge acquisition, this quality is valuable.

In [43] (1991) we explore this ability. Each node of the MLP seems to learn a particular characteristic of the firm in a way somehow similar to the procedure used in ratio analysis.

A second cause for the good generatisation of the MLP occurs when the minimum number of nodes in any hidden layer matches the number of features important for the relationship to be learned, the likelihood of each one of those nodes becoming a model of a different feature of such relation is larger. If that happens, the MLP performs an effective features extraction. As a consequence, the generalisation capacity receives a further improvement.

Some applications take advantage of the trend towards individuality the nodes in the MLP exhibit, to find the important features of a set of examples. Using a topology known as the Bottle-Neck MLP, the same set of examples are presented both as input and as

outcome. This technique is similar to Factor Analysis. But when more than one hidden layer is used, also non-linear features are extracted. A Bottle-Neck MLP has been used to test the Arbitrage Pricing Theory.

Iterative optimization: A Multi-Layer Perceptron adjusts the free parameters which ought to model a relation in small steps. Each of these steady improvements seek an advance in the minimization of the observed deviations between the produced output and the desired outcome. Therefore, the learning of a relation progresses steadily along many small steps. This allows a broad manipulation of the free parameters — known as weights — engaged in the building of the model. Such a manipulation, unavailable in analytical tools, turns out to be essential for achieving good generalisation and interesting internal representations.

Tight control over the modelling power: Another important characteristic of the MLP is that a tight control can be attained over the flow of information for modelling a given relationship as well as over the amount of non-linearity introduced.

The number of nodes in any layer determines the maximum dimension of the modelled relation. For example, by using a hidden layer with three nodes, we constrain the relation to be modelled to have three dimensions. On linear grounds this would mean that the matrix representing the desired relation would have one of its dimensions set to three.

This fact allows a direct control over the power of an MLP when performing classification with non-linear boundaries. Since classification with arbitrary boundaries requires the artificial enlargement of the dimension of the input space, by controlling it we define exactly the kind of boundaries we allow for classifying.

Also the amount of non-linearity an MLP apportion to the model depends on the total number of nodes in the hidden layers, no matter its topology. We can have two hidden layers, each one with two nodes, or one unique hidden layer with four nodes. The amount of non-linearity allowed would be approximately similar in both cases.

Together, the last two characteristics of the MLP make it remarkably flexible in the use of its power. Hence, the MLP is flexible in its generalisation as well.

Easy implementation of optimization and convergence criteria: Finally, when appropriate, we can easily change the optimization criterion for it is independent of the optimization process. For example, Minimum Least-Squares deviation, as a success measure, is just one of the possible criteria. Likelihood maximization seems more appropriate for problems involving classification. In such a case, the Multi-Layer Perceptron learns to maximize

the probability of having obtained the set of input-output pairs which were actually observed in the training set. This flexibility adds up to the MLP's already good one.

Also, the learning itself can be carried out using simply the generalised delta rule or more elaborated stochastic optimization techniques involving, for example, simulated annealing. In general, the possibility of simply acting upon the rate of convergence and the individual increments each parameter receives during the learning process can be most valuable for achieving meaningful internal representations.

3.4 Discrete Versus Continuous-Valued Outcomes

An MLP can perform either non-linear multi-variate regression or non-linear discriminant analysis. In the first case it is called upon for approaching a continuous-valued outcome. In the second one, outcomes are discrete states. The correct classification of classes in the statistical exclusive-OR problem is an example of the second kind of task.

When the MLP is used as a classifier, the number of nodes and layers employed is more critical than when it is used in regressing. In the former case, the MLP must have enough nodes to form the frontiers required by the modelled problem.

Each first-hidden-layer node creates a hyper-plane in the input space since its input is a linear combination of input variables. In the layer next to this, several of these hyper-planes can be used to define a region enclosing a particular group of examples relating to one of the given outcomes. That is, the first hidden layer needs to have as many nodes as linear pieces required to build the frontiers for separating groups, and the second layer needs to have as many nodes as different groups. When inputs and outcomes are only statistically separable, that is, when similar input vectors in the learning set relate to different groups, the final error of the net, after convergence, cannot be zero. The above reasoning holds, but now we should envisage probability surfaces (gradients of likelihood) instead of well-defined, deterministic, frontiers.

Also, when using the MLP as a classifier, it's more reasonable to substitute the described minimum Least-Squares Error criterion by a more appropriate one. For example, Likelihood maximization is often used. In that case, the model selected is the one which maximizes the probability of having obtained samples as those actually used in the learning set. When the number of nodes in the last layer matches the number of groups to be classified, it can be shown that an appropriate coding of outcomes makes the MLP output directly the probability of obtaining such an outcome given such inputs. Solla *et al.* [38] contains the appropriate formalism.

When outcomes are continuous-valued, no transfer function should be used in the last

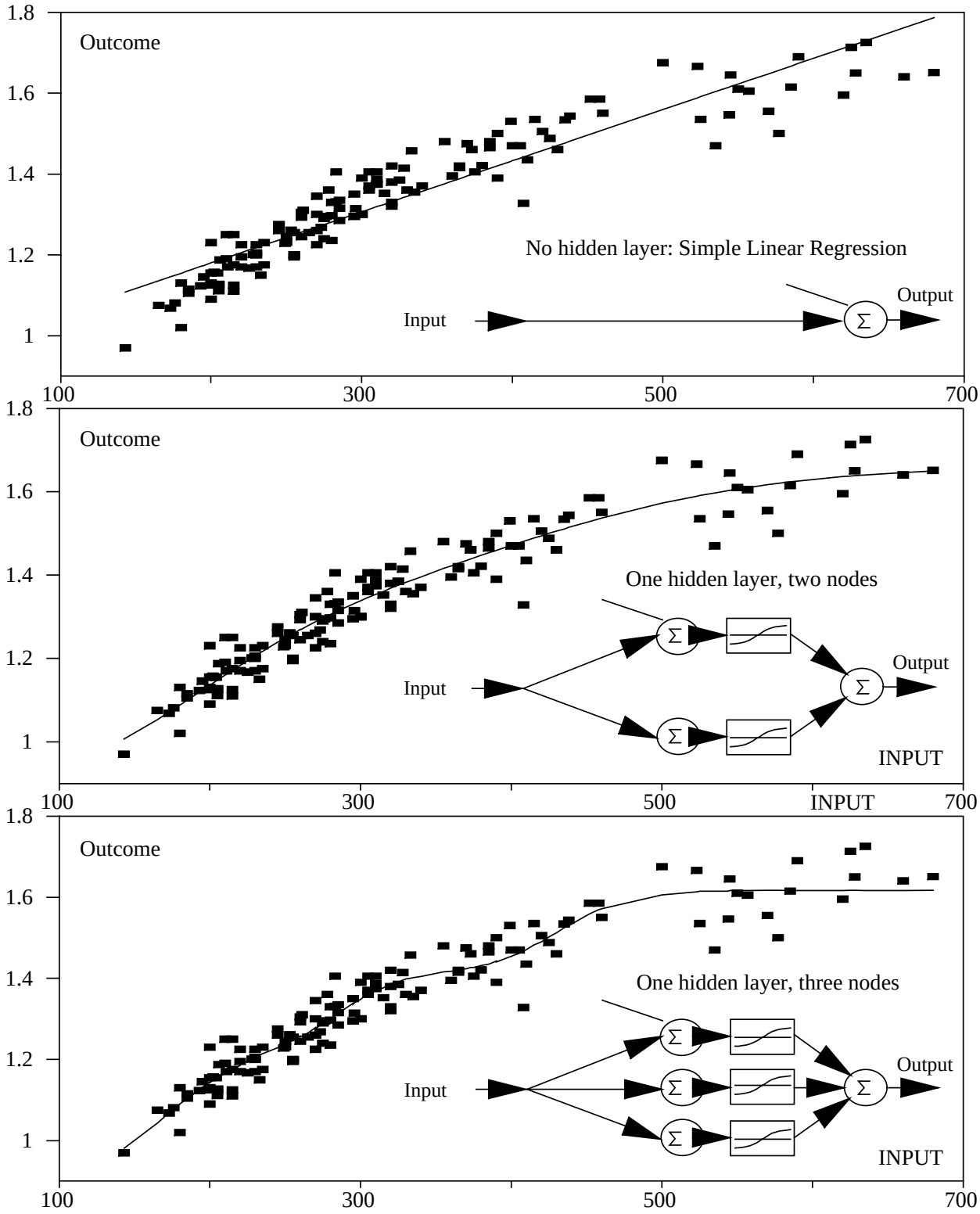

Figure 7: The MLP mimicking a regression: It approaches a bivariate relationship by apportioning a controlled amount of non-linearity. This amount depends on the number of nodes.

layer of nodes since it would limit the range outputs can attain. It is also important to bear in mind that the amount of non-linearity introduced is controlled by the total number of nodes having transfer functions, regardless of its position. An exaggerated number of nodes will produce a too detailed — and hence very sample-dependent — model. Figure 7 shows the effect, in a simple case (one unique input and output) of introducing more nodes in the MLP.

In the case of continuous-valued outcomes the appropriate success criterion is the minimization of the Least-Squares Error. To control the learning process it is common practice to use the overall R^2, that is, the proportion of the variability of the targets explained by the model, corrected for the number of free parameters engaged.

3.5 The Delay-Line MLP for Forecasting

An important consequence of the Wiener-Volterra analysis is that, under very general circumstances, it is possible to model the internal behaviour of any system just by using two feed-forward steps. One linear step incorporating a certain amount of memory of past events, followed by a simple non-linear map (see Schetzen 1980 [32]). This result is equivalent to saying that we can mimic complex mechanisms like systems of non-linear differential equations or a chaotic attractor just by pulling together a linear filter and a non-linear function.

Lapedes and Farber (1987) [16] used this principle to show that an MLP was suitable for performing systems identification, time-series prediction and similar tasks. They simply used the input vector as a delay-line (see figure 1). Hence, the first hidden layer acts like a Wiener filter. Subsequent layers introduce the required amount of non-linearity.

When used in this fashion, an MLP becomes a very effective predicting tool. Its generalisation, flexibility and ease of use makes it substantially more attractive than the equivalent analytical procedures. Especially in the prediction of time-series apparently complicated but with an underlying dynamic mechanism, the use of these delay-line MLP often mean a decisive improvement.

We used delay-line MLP to identify the underlying system governing a few random number generators. The resulting topology was very simple. We also tested their use in the modelling of first-difference chaotic series observing good predicting performances. Refenes

Notice that most of the tasks referred to here could be more elegantly performed using recurrent algorithms instead of delay-lines. Recurrent networks engage a smaller number of weights than delay-line networks. Therefore they achieve a better generalisation and require smaller learning sets. Their learning is also faster. Recurrent networks can also cope with missing values in the learning set. In the presence of a missing value, they provide the most

likely value in the context. This feature is typical of Hopfield networks and is known, in its more general form as "Associative Recall". However, the use of recurrence requires a more advanced practice and its exploring is not adequate as an introduction to Neural Networks. Almeida (1987) [1] explains this formulation.

3.6 The Modern MLP

The characteristics which make modern MLP different from the original algorithm (Rumelhart *et alt.* 1986 [31]) can be summarized as:

- Cross-Validation of results.
- Incomplete training.
- Pruning of weights or nodes.
- Learning rates particular to each weight.

We now comment on these characteristics. The first one relates to improvements in the ability to generalise. It is a particular implementation of a known procedure, the Cross-Validation (see Stone 1978 [39]). In order to obtain an estimate of the generalisation capacity of a model, the original samples were divided randomly into two sub-samples. Models were constructed with one sub-sample, the training set, and a check carried out with the other one, the test set. When the outcomes are continuous-valued, the adequate measure of quality of fitting is the proportion of explained variability, R^2, corrected for the degrees of freedom engaged. The performance of the MLP on the set used for training depends solely on the number of free parameters and can be increased simply by introducing more nodes on the net. Therefore such results are uninteresting.

Since the MLP seeks an optimum iteratively, we can stop its training when an optimum is obtained in the test set rather than in the training set. We thus prevent this powerful algorithm from over-fitting the data. The Back-Propagation algorithm seeks the modelling of progressively smaller or less important features of the relation, during the learning process. Firstly, broad features are accounted for: The mean, a linear trend. Then, more detailed ones are modelled. Hence, the effective degrees of freedom the MLP engages can be viewed as increasing during learning. Assuming that the topology of the net contains plenty of free parameters, the MLP will be able to model, not only the desired features but also the undesirable random uniqueness of a particular sample. We prevent it from doing this by stopping the process before finishing. The appropriate moment for stopping is when

the results, as measured by the test set, are optimal. For a good topology, the fact that the learning stops before a minimum is reached in the learning set clearly enhances generalisation.

Finally, each one of the weights in the modern MLP has its own increment, adjusted as described in Silva and Almeida (1990) [37].

Also, some methods for pruning the MLP are used. Amongst those, "Skeletonization" [25] and "Optimal Brain Damage" [17] are the most popular. Notice that the first one is intended to reduce the number of nodes, not the one of weights. A very simple method for pruning the useless weights in the MLP consists on introducing, during training, a penalization of very small weights. If the input variables were very differently scaled, small weights could just mean that the MLP was trying to scale down a particular variable. But when the input to the MLP is mean-adjusted and have similar standard deviations, the only reason for any some weights to remain small throughout the learning is to try to diminish the importance of one variable to the output of the node it belongs to. In the MLP, each node acts as a modelling unit with a certain amount of free parameters. The same output can be obtained with very different combinations of weights. If we introduce a penalization of very small weights during the training, as the correction of weights is proportional to the input variables, small weights tend to remain small. In the same way, large weights tend to have their values strengthen. The final result is a contrasted set of weights: The first layer now contains only very large or very small weights. The information concerning the modelled relation is concentrated in a few weights instead of distributed by all of them [43].

4 Self-Organized Neural Maps of Patterns

Statistical modelling tools are often used to describe functional relationships. However, in some cases we want to build a representation of a density of cases, not the one of a relationship. These representations are known as *topology-preserving maps*. The reasons for using maps can be twofold: Either the information regarding the position of each case in a distribution is relevant and must be preserved, or the density of cases draws a shape which is not simple enough for being approached by the usual functions.

The Kohonen's Self-Organized Map of Patterns is a fast and simple heuristic able to reproduce an original density using a small number of points that somehow are an image of that distribution. It consists of a lattice of nodes, each of them containing its own set of adjustable weights: In the j^{th} node, a weight vector $W = w_{j1}, w_{j2}, \ldots, w_{jM}$ links to a corresponding input vector $X = x_1, x_2, \ldots, x_M$. Each node's output is a function of both

the input and its weights: $o_j = f(X, W)$. Two frequently used output functions are

$$o_j = \sum_{i=1}^{M} x_i \times w_{ji}, \quad \text{an inner product, or} \quad o_j = \sqrt{\sum_{i=1}^{M} (x_i - w_{ji})^2} \quad \text{an Euclidean distance.}$$

It is a requirement that these functions measure the distance or similarity between W and X. The training of the map takes place as follows: All the nodes are supplied with the same input vector, extracted at random from the learning set. Then, the node with the largest output is discovered. It will be the one whose vector of weights shows the greater similarity towards the presented input. Next, a neighbourhood is defined around this node, and the weights of all nodes in such a neighbourhood are updated or "rewarded" in a way that makes them more similar to the observation they identified. For example, the new value of a weight, w_{ji}, linking the input i with a rewarded node j, can be calculated as

$$w_{ji}^{t+1} = w_{ji}^{t} + \eta \times (x_i - w_{ji}^{t})$$

in which t and $t+1$ denote a sequence and η is a small increment. The nodes not in this neighbourhood receive no rewarding. The procedure is repeated for all the vectors in the learning set and then again and again. At length, specific nodes become "excited" by particular inputs so that the topological relationship between them is mirrored by the position of the nodes. We refer to any node within a lattice by saying that the integer m is a counter of the rectangle's row number and the index n is a counter of its column number. Nodes are determined by a pair $\{m, n\}$. After training, the map will show, for each new input, a pair $\{m, n\}$ identifying the node that it excites. That is, we mapped a continuous-valued space onto a discrete one, but preserving the original topology.

Clearly, the heuristic explained above is a variation of the Hebb's Rule. Kohonen's maps owe more to Connectionism than the MLP. In the last one, learning is inspired in engineering practice rather than in the brain.

The number of nodes and its form is determined prior to the learning. This topology must be prone for adopting the shape it is set to learn. Figure 8 on page 29 shows the positions, after the learning process has finished, of a lattice of nodes superimposed to the density of cases they reproduce. Straight lines link nodes that are neighbours. The map in figure 8 is an example of the use of Kohonen's heuristic to build a diagnosis table relating the m and n of a fired node to profitability and the magnitude of its components inside firms. Such tables could be more or less detailed depending on the number of nodes used. If, in our example, we were to use a larger number of nodes in the n dimension (for example, 5 instead of 3), we would get more specific diagnostics. But it is not clear whether such an increase would be desirable.

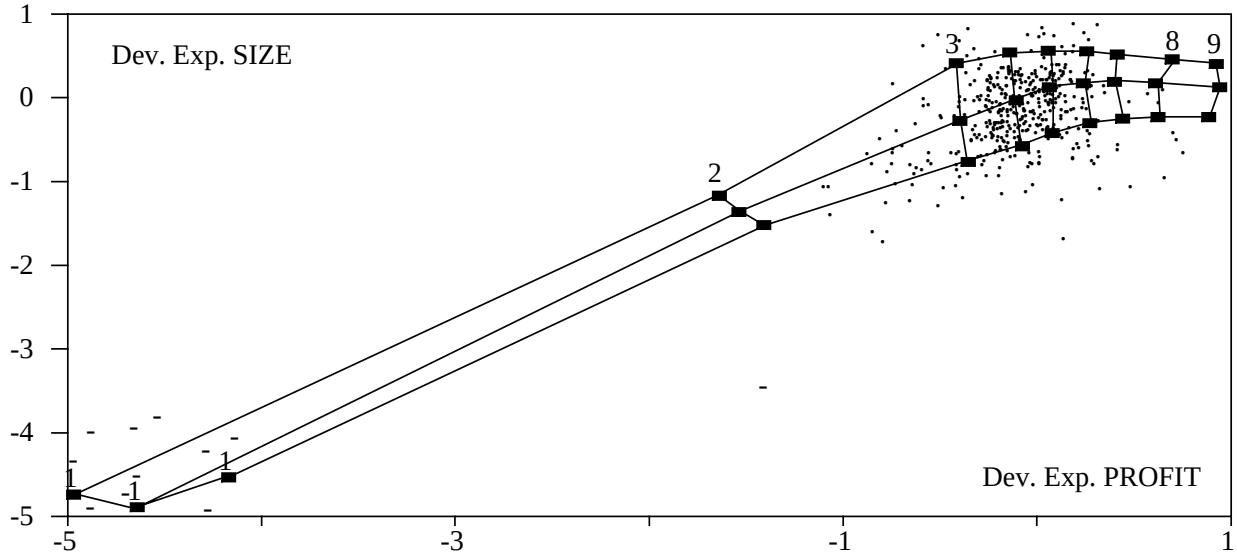

Figure 8: The lattice of 9×3 nodes superimposed to the RRP whose density of cases they have learned. Nodes that are neighbours have been linked by solid lines.

Self-organized maps are adequate to perform tasks such as:

- Dimension reduction and Intrinsic dimension assessment: Find an f such $f : \mathbb{R}^N \mapsto \mathbb{R}^M$, $(N > M)$, having some optimal quality, or find the smallest $M < N$ for which an f exists such $f : \mathbb{R}^N \mapsto \mathbb{R}^M$ having some optimal quality. This is because in Kohonen's maps it is unlikely to obtain an ordered map when using nodes forming a lattice of smaller dimension than the intrinsic dimensionality of the training set. For example, if a two-dimension lattice of nodes is used to map a really three-dimensional density of cases, the weight vectors will fold in waves, attempting to fully cover the 3-D space. In this case, no real relation exists between inputs and the density function. Therefore, folding can be used as a diagnostic for tracing an over-reduced dimension reduction attempt. Procedures are available to discover folding in more than 3-D maps. Recently, Serrano *et al.* (1993) [34] used Kohonen's maps to reduce the dimension of several financial measures onto a two-dimensional lattice, obtaining promising results.

- Mapping as pre-processing for further MLP classification or for expert systems. Once an intrinsic dimension has been recognized and a map produced, an MLP can be used to classify discrete outputs according to features. This procedure would be the connectionist equivalent to Factor Analysis prior to Discriminant Analysis. Also, the discretization obtained with Kohonen's maps can be useful to transform continuous-valued data, like the one found in financial reports, onto diagnostic tables suitable for expert systems.

- Tracing dynamic features (like trajectories): When input vectors are successive events, self-organized maps will draw a trajectory, should the variables bear any kind of joint trend (cross-correlations not zero). Several different trends can be identified by their trajectories. In the above example, if we input the map with a sequence of vectors representing the same firm during several periods of time, it will output the corresponding sequence of fired nodes. This output sequence defines a trajectory in the discrete space of the lattice of nodes.

A drawback of Kohonen's heuristic is that sometimes it is difficult to make the map span the whole of the scatter. Cases may occur that will not fire any node at all. But this happens mainly when, during the training of the map, the neighbourhood for nodes is defined in a simplistic way. Any attempt to reproduce Kohonen's heuristic should use more elaborated definitions of neighbourhood than those given in introductory texts.

Besides Kohonen's Self-Organized Maps, other algorithms exist able to perform the same task. Some of the tools known as "quantizers" could also be used to to obtain maps. However, this heuristic is simple to implement and to explore, and its training is fast. It also illustrates a practical use of the Hebb's Rule. However, Competitive Learning and, in general, the Connectionist approaches, can have some unexpected qualities: They are local and parallel in structure, which makes them ideal for implementation on future machines; and they are robust regarding assumptions about the data.

5 NNs in Finance and Accounting Research

Neural Networks are not a key to all kinds of data-analytical problems. They offer some specific advantages and they have their own drawbacks. This section mainly focus on those fields in which we think that their use is advantageous. The examples explored in this study illustrate important and promising capabilities of Neural Networks in accounting and finance research.

NNs present two drawbacks: Their training isn't straightforward, and the resulting models often are neither interpretable nor portable, acting like "black-boxes".

The difficult training of NNs has many manifestations. Firstly, it requires an exorbitant amount od CPU, especially when the number of examples is large. In practice, this task isn't properly undertaken by the usual PC, even the fastest. Secondly, the training isn't, in general, automatic. It requires supervision from the operator, and some practice. This is another reason for avoiding slow machines. Thirdly, the number of nodes or layers to be used in every case remains a question of empirical testing, despite the large amount of literature

devoted to providing some guidance in the selection of an adequate topology. Ultimately, only by testing can we ascertain about the fittness of a given Network. Finally, there are plenty of small, undocumented, tricks, important for avoiding local minima, for speeding-up the convergence, or for obtaining a parsimonious model.

The second drawback, lack of interpretability, must be seen in the light of what we get by using NNs. It's clear that the most complicated models produced by NNs cannot be interpreted by direct observation. However, this is because the modelled relationship is itself complex. Modelling aims at finding the simplest, satisfactory, map. If the trainer of a network is not indulging in building unnecessarily complicated models, the resulting model is as complex at it should be, and probably less complex than other models obtained by different tools when performing the same task. The fact that the usual NNs don't produce rules or simple equations when solving problems hitherto unsolved, simply means that rules or simple equations are inadequate for capturing those relationships.

It must be said that, in accounting and finance research, NN models often are interpretable and even revealing. This is because the kind of relationships to be modelled isn't too complex.

There are one or two pitfalls awaiting those engaged in using Neural Networks, especially if their practice in Statistical Modelling isn't rooted in common-sense. These pitfalls stem from the power of NNs and aren't present when using linear tools. Firstly, Cross-Validation must never be omitted when modelling with NNs. Secondly, even when cross-validating the results, the topology of the network must be parsimonious. When the number of examples in a training set is small, the network must also have a small number of weights in the first hidden layer. Broadly, no more than five weights for every hundred. Thirdly, the performance of a model must be discussed in the light of its *robustness* and *economy*, not in isolation. For example, we must study the relationship between the number of weights in several alternative models, and their performance, selecting the one that makes the best of its degrees of freedom, not the one that performs the best. Finally, several sets of random weights must be used when training a given model. It's frequent to obtain, when using a given set of initial weights, a final performance that is much better or much worse than the one obtained with another set of weights. This is because the optimum, when training the NN, is obtained by search. Therefore, the possibility of finding a local optimum, not the absolute one, is real. When using NNs, we can never be sure that the optimum we found is the best. After obtaining a model that performs not too badly, it's frequent to discover other ones that perform much better, just by using a different set of initial weights.

In general, Cross-Validation is enough to ensure that a model is parsimonious: In order

to obtain a good performance in the test set, the model has to be be pruned to the limit. However, we must be aware that, in occasions, unnecessarily complicated NNs are capable of a surprisingly good generalisation. This generalisation isn't to be trusted because it probably isn't robust.

We now summarize the tasks NNs are called upon to perform.

Classification: The most obvious application of the MLP is in classification. Discriminant Analysis can establish linear or quadratic boundaries between groups. This seems enough in the majority of current problems. A Multi-Layer Perceptron will draw boundaries of any shape. Thus, it is able to cope with complex relations involving higher order effects. But not only this. An MLP models complex relations in a very parsimonious way (the number of free parameters engaged in the modelling can be optimized) and the non-linearity used for defining frontiers is local.

Bond rating and lending decision mimics have already been attempted with the MLP [4]. A recent study on the selection of Neural Network architectures for improving generalisation uses bond rating as an example [44]. See also [9] for a review of some Neural Net applications being developed by a specialized firm. Firm distress prediction has also been modelled by an MLP. This, despite the problem being well suited for linear algorithms.

The described applications are just direct extrapolations of classic and thoroughly explored problems in accounting research. They simply substitute the linear techniques by the MLP. We think that such experiments are not the most adequate way of showing the real possibilities of Neural Networks since there is very few of specifically related to Neural Networks on them. Instead, we centre this study in what Neural Networks can do and the other tools can't.

Assessment of dimensionality and dimension reduction: Statistical tools like Multiple Discriminant Analysis or Factor Analysis are unable to clearly point out the intrinsic dimensionality of the data. N input variables or groups lead to N factors or $N-1$ scores. Despite the use of some ad-hoc tests, it is after all the intuition of the researcher who decides how many of these dimensions are to be considered as real features. Possibly, guesses of intrinsic dimensionality of processes like economic pervasive factors influencing capital markets or basic common sources of variability in ratios, based as they are on conjectures about acceptable uniqueness, are over-estimations.

One of the most promising applications of self-organizing maps of patterns lies in the fact that the real dimensionality of the data is recognized as a basic characteristic [14]. The real dimension of market expected returns or accounting statements could then be assessed.

Also the discrimination between different kinds of firm distress — should they exist — could benefit from the capacity of Kohonen maps to trace dynamic features. Since the relation between accounting data and the outcome is in this case very strong — the outcome can be predicted with small confidence limits — it seems as if a simple self-organized map of patterns would be enough to trace it. Such a map would also be able to discriminate between trajectories leading to insolvency.

Tracing Dynamic Features Self-Organized Maps of Patterns are also useful in capturing positions and trajectories of firms drawn in the space of their financial features. This technique can be a step towards the automation of ratio analysis.

In [42] we use Self-Organized Maps of Patterns and two-dimensional ratios to improve the financial analysis of firms. Our framework is aimed at an automatic exploring of large databases containing accounting data. It can be seen as a pre-processor, able to bridge the gap between continuous-valued, stochastic, data like the one of accounting reports, and symbol-based expert systems. The information extracted with this technique can be fed into more general expert systems along with other sources of knowledge. For example, several of the developed tools could cover each one a particular feature of the firm — liquidity, capital turnover and so on — outputting rules that would be jointly processed by an expert system.

Our tool allows both cross-sectional positions of individual firms and trajectories during a time period to be traced. It is close to ratios, yielding similar diagnostics: Whether a particular firm is above the standards, near the expected or below the standards. But this, referred to two aspects of an accounting feature, not just one at a time. For example, whilst ratios can assess liquidity either as a contrast between the amount of current assets and the one of current liabilities or, alternatively, as a contrast between the amount of working capital and the size of the firm, our tool can show both aspects of liquidity together.

The quality of the diagnosis and its interest rely on the selected variables as happens with ratios. It is the experience of the analyst which dictates which items are to be used and how to interpret the resulting maps. Since these tools are close to ratios, all the expertize of ratio analysis can be directly implemented on them.

Features Extraction: After the assessment of the intrinsic dimensionality, an MLP can construct new variables containing the main sources of variability present in the data. These factors can be built so as to be similar to Principal Components or, alternatively, to capture non-linear features. In the later case, the extracted factors are also representations of the data, extracted in such a way that the average missing information becomes minimal. But not subjected to the condition of being linear.

In other words, Neural Networks can, if required, extract non-linear pieces of information from the multi-variate distribution. For example, if in some two-dimensional phenomena its scatter diagram shows a clear "S" shape, the first or main feature to be extracted can be the S shape itself. An MLP will classify other S-shaped distributions as sharing that feature with the original data. This can be decisive when trying a classification of sensitivities of assets to market forces based on accounting reports and related information, or other situations where linearity doesn't apply.

Forecasting and Systems Identification: The MLP is suited for forecasting as well. In this case, the desired outcome is the same time-history as the input but placed a few periods ahead. Each input variable is related to the others so that the information fed into the MLP is a window representing a given time period. The learning takes place by showing the net many of these windows selected at random, along with the corresponding time-history a desired number of periods ahead. As a result, the MLP learns to predict the underlying phenomenon.

A description of the MLP in forecasting and Systems Identification can be found in Lapedes and Farber (1987) [16]. White [46] used an MLP to try to predict the returns of common stock.

Systems Identification is potentially interesting for assessing the extent to which some financial time-histories are dictated by a complex chaotic behaviour rather than by simple randomness. It is possible that the price of some commodities are a non-linear dynamic one as well. Benoit Mandelbrot noticed one such structure in the price of cotton [20] and other authors suggested similar behaviour in indices related to equity in the NYSE [26]. If that is so, the MLP would be a most adequate tool for capturing the underlying mechanism.

Dynamic Features, Stability and Diagnostics: Neural Networks can even cope with time-varying multi-variate data patterns as a whole. Time correlations of non-stationary data can carry important information about underlying trends.

Considering each M-dimensional cross-section input as a vector, if there is some relation between an event and its predecessors, a trajectory of such a vector will be drawn in $\mathbb{R}^M$. This trajectory can be recognized by an MLP or, in some cases, by a self-organized map of patterns after appropriate reduction. See Tattersall (1988) [40] for an explanation of this technique.

This seems a promising diagnostics tool for discussing the stability of APT factors, different kinds of firm distress or ratio information contents.

6 Summary

Neural Networks are versatile modelling tools likely to become useful in specific problems involving the extraction of knowledge from samples of accounting and financial data. The more promising tasks seem to be those related to:

- Complex classification and systems identification. In this case the valuable feature of Neural networks is their power and generalisation capacity. In [41] we explore this aspect.
- Features Extraction via node's specificity. In this case the feature viewed as interesting is the information apportioned by the model about the intrinsic structure of the data.

Neural Networks embody two main sources of inspiration: Connectionism and Tele-Communications Engineering. Examples have been given of the most representative networks in both cases: The Multi-layer Perceptron and the Self-Organized Map of Patterns.

References

[1] L. Almeida. A learning rule for asynchronous perceptrons with feedback in a combinational environment. In M. Caudil and C. Butler, editors, *Proceedings of the First IEEE International Conference on Neural Networks*, 1987. San Diego, California, June.

[2] J.A. Anderson. Two models for memory organization. *Mathematical Biosciences*, 8:137–160, 1970.

[3] R. Booton. Nonlinear control systems with stochastic inputs. Technical Report 61, MIT: Dyn. An. and Contr. Lab., 1952.

[4] S. Dudda and S. Shekhar. A non-conservative application of neural networks. In R. Trippi and E. Turban, editors, *Investment Management, Decision Support and Expert Systems*, pages 271–282. Boyd and Fraser, 1990.

[5] R. Forsyth. Ai: East / west conference report. *AISB Quarterly*, 66:14–17, 1988.

[6] D. Gabor, W. Wilby, and R. Woodcock. A universal non-linear filter, predictor and simulator which optimizes itself by a learning process. *Journal of the Institution of Electrical Engineers*, pages 422–435, 1960.

[7] S. Grossberg. *The Adaptive Brain*, volume 1–2. Elsevier, Amsterdam, 1987.

[8] J. Hebb. *The Organization of Behavior.* Wiley, 1949.

[9] R. Hecht-Nielsen. Neurocomputer applications. In *Neural Computers*, pages 445–453. NATO ASI Series, Serie F, Number 41, 1987. Edited by R. Eckmiller and C. Malsburg in Springer Verlag, Berlin.

[10] G. Hinton and T. Sejnowski. Optimal perceptual inference. In *Conference on Computer Vision and Pattern Recognition*, pages 448–453. IEEE, Washington, DC, 1983.

[11] J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Science USA*, page 2554. USA Academy of Science, 1982. Volume 79.

[12] I. Kasakov. Approximate probability analysis of the operational precision of essentially nonlinear feedback control systems. *Automation and Remote Control*, 17, 1956.

[13] T. Kohonen. An adaptive associative memory principle. *IEEE Transactions on Computers*, C-23:444–445, 1974.

[14] T. Kohonen. *Self-Organization and Associative Memory.* Springer Verlag,Berlin, 1984.

[15] A. Kolmogoroff. *Interpolation and Extrapolation of Stationary Series.* Academy of Sciences of the URSS, 1942.

[16] A. Lapedes and T. Farber. Nonlinear signal processing using neural networks. Technical Report LA UR 87 2662, Los Alamos National Laboratory, 1987.

[17] Y. LeCun. Optimal brain damage. In *Neural Information Processing Systems*, volume 1. Morgan Kaufmann, Denver 1990.

[18] Y. W. Lee. Contributions of norbert wiener to lineasr and non-linear theory in engineering. *Selected Papers of Norbert Wiener*, 1:17–34, 1964.

[19] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, 4:4–22, 1987.

[20] B. Mandelbrot. The variation of certain speculative prices. *Journal of Business*, 36:394–419, October 1963.

[21] W. McCulloch and W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Boulletin of Mathematical Biophysics*, 5:115–133, 1943.

[22] W.S. McCulloch. *Embodiments of Mind.* The MIT Press, 1965.

[23] J. Mingers. Rule induction with statistical data — a comparison with multiple regression. *Journal of the Operational Research Society*, 38(4):347–351, 1986.

[24] M. Minsky and S. Papert. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1969.

[25] M. Mozer and P. Smolensky. Skeletonization: A technique for trimming the fat from a network via relevance assessment. In *Neural Information Processing Systems*, volume 1. Morgan Kaufmann, Denver 1988.

[26] E. Peters. Fractal structure in the capital markets. *The Financial Analysts Journal*, pages 32–37, July 1989.

[27] W. Pitts and W.S. McCulloch. How we know universals: The perception of auditory and visual forms. *Bulletin of Mathematical Biophysiology*, 9:127–147, 1947.

[28] J. Quinlan. Discovering rules from large collections of samples — a case study. In D. Michie, editor, *Expert Systems in the Micro Electronic Age.* Edimburgh University Press, 1979.

[29] P. Race and R. Thomas. Rule induction in investment appraisal. *Journal of the Operational Research Society*, 39(12):1113–1123, 1988.

[30] F. Rosenblatt. *Principles of Neurodynamics.* Spartan Books, Washington, 1961.

[31] D. Rumelhart, G. Hinton, and R. Williams. Learning internal representations by error propagation. In *Parallel Distributed Processing*, volume 1. The MIT Press, 1986.

[32] M. Schetzen. *The Volterra and Wiener Theories of Non-Linear Systems.* Wiley, 1980.

[33] T. Sejnowski and C. Rosenberg. Nettalk: A parallel network that learns to read aloud. In *Electr. Engin. Dept.* The John Hopkins University, 1986. JHU/EECS-86/01.

[34] C. Serrano. Artificial neural networks in financial statement analysis: Ratios versus accounting data. Technical report, European Accounting Association Annual Meeting, Turku, Finland, 1993.

[35] D. Shen and A. Rosenberg. A nonlinear stochastic learning model. In *Trans. of the third Prague Conf. on Information Theory, Statistical Decision Functions and Random Processes*, pages 629–636. Czekoslovak Academy of Sciences, 1964.

[36] W. Shen. Nonlinear amplitude sensitive control systems with stochastic inputs. In *WESCON Convention Record*. IRE, 1957. part 4.

[37] F. Silva and L. Almeida. Acceleration techniques for the backpropagation algorithm. In *Neural Networks*. EURASIP Workshop, Sesimbra, Portugal, 1990. L. B. Almeida and C. J. Wellekens (Eds.), Springer-Verlag.

[38] S. Solla, E. Levin, and M. Fleisher. Accelerated learning in layered neural networks. *Complex Systems*, 2:625–640, 1988.

[39] M. Stone. Cross-validation: A review. *Math. Operationsforsch. Statist., Ser. Statistics*, 9(1), 1978.

[40] G. Tattersall, P. Linford, and R. Linggard. Neural arrays for speech recognition. *Br. Telecom Technol. J.*, 6:140–163, 1988.

[41] D. Trigueiros. A taxonomy of risk in large uk industrial firms. In E. Nunamaker and R. Sprague, editors, *Organisational Systems and Technology: Proceedings of the* 26^{th} *Hawaii International Conference on System Sciences*, pages 587–596. IEEE Computer Society Press, Los Alamitos, CA, 1993.

[42] D. Trigueiros. Improving ratio analysis with maps of patterns. In E. Nunamaker and R. Sprague, editors, *Organisational Systems and Technology: Proceedings of the* 27^{th} *Hawaii International Conference on System Sciences (Forthcoming)*, pages —. IEEE Computer Society Press, Los Alamitos, CA, 1994.

[43] D. Trigueiros and R. Berry. The application of neural network based methods to the extraction of knowledge from accounting reports. In E. Nunamaker and R. Sprague, editors, *Organisational Systems and Technology: Proceedings of the* 24^{th} *Hawaii International Conference on System Sciences*, pages 136–146. IEEE Computer Society Press, Los Alamitos, CA, 1991.

[44] J. Utans and J. Moody. Selecting neural network architectures via the prediction risk: Application to corporate bond rating prediction. In IEEE Computer Society Press, editor, *Proceedings of the* 1^{st} *International Conference on Artificial Intelligence Applications on Wall Street*, 1991. Los Alamitos, California.

[45] V. Volterra. *Theory of Functionals*. Blackie, 1930.

[46] H. White. Economic prediction using neural networks: The case of ibm daily returns. In R. Trippi and E. Turban, editors, *Investment Management, Decision Support and Expert Systems*, pages 283–292. Boyd and Fraser, 1990.

[47] B. Widrow and E. Hoff. Adaptive switching circuits. In *WESCON Convention Record.* IRE, 1960. part 4.

[48] N. Wiener. *The extrapolation, intrapolation and Smoothing of stationary time-series.* Wiley, 1949.

[49] N. Wiener. *Nonlinear Problems in Random Theory.* Wiley, 1958.

[50] N. Wiener. *Cybernetics.* The MIT Press, 1961.