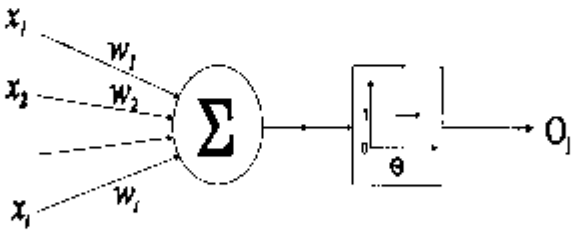
Anthony W. Minns

UNESCO-IHE
Institute for Water Education

Artificial Neural Networks as Subsymbolic Process Descriptors



ARTIFICIAL NEURAL NETWORKS AS SUBSYMBOLIC PROCESS DESCRIPTORS

Artificial Neural Networks as Subsymbolic Process Descriptors

DISSERTATION
Submitted in fulfilment of the requirements of
the Board for the Doctorates of Delft University of Technology
and the Academic Board of the International Institute for Infrastructural,
Hydraulic and Environmental Engineering for the Degree of DOCTOR
to be defended in public
on Monday, 2 March 1998 at 13:30 h

by

ANTHONY WILLIAM MINNS
Bachelor of Engineering (South Australian Institute of Technology)
Diploma in Hydraulic Engineering (IHE Delft)
born in Adelaide, Australia

This dissertation has been approved by the promoter:

Em. Prof. dr M. B. Abbott	Delft University of Technology/International Institute for Infrastructural, Hydraulic and Environmental Engineering

Other committee members:

Chairman:	Rector Magnificus, Delft University of Technology
Co-chairman:	Rector, International Institute for Infrastructural, Hydraulic and Environmental Engineering
Prof. dr M. J. Hall	International Institute for Infrastructural, Hydraulic and Environmenal Engineering, Delft
Prof. dr ir P. van der Veer	Delft University of Technology, Delft
Prof. dr D. P. Loucks	Cornell University, Ithaca
Prof. dr P. E. O'Connell	University of Newcastle-Upon-Tyne, Newcastle
Prof. dr P. Goodwin	University of Idaho, Boise

Published by
A. A. Balkema, P.O. Box 1675, 3000 BR Rotterdam, Netherlands
Fax: +31.10.4135947; E-mail: balkema@balkema.nl; Internet site: http://www.balkema.nl

A. A. Balkema Publishers, Old Post Road, Brookfield, VT 05036-9704, USA
Fax: 802.276.3837; E-mail: info@ashgate.com

ISBN 90 5410 409 0

Printed in the Netherlands

to René,
and in loving memory of Marijke

-

Contents

Abstract

Hydroinformatics proceeds into that which M.B. Abbott has characterised as the 'post-symbolic' era along two different paths. Along the one path, it elaborates tools and even more general working environments for engineers, environmentalists and other professionals that make little or no use of symbols in any conventional sense, but which instead work almost entirely with signs. Along the other path, as illustrated in this present work, hydroinformatics increasingly uses non-symbolic, and indeed strictly sub-symbolic, methods in order to construct these tools and more general working environments. Of course, in this latter case, the constructor of these instruments must still make recourse to symbolic representations, but, as explained here, these are employed essentially as aids to the thinking processes of the constructor, and are not carried over or incorporated in any way into the operations of the constructions themselves. It is this second path of sub-symbolic constructions that forms the subject of the present work.

The first chapter of this work is given over to an introduction to the sub-symbolic paradigm in general and within the particular context of hydroinformatics. The three main current divisions within the sub-symbolic paradigm are those of artificial neural networks (ANNs), evolutionary algorithms and cellular automata. A detailed description of artificial neural networks and, in particular, feed-forward, multi-layer perceptrons, which constitute the sub-symbolic tool used throughout this work, is given in Chapter 2. A brief description of the other two sub-symbolic methodologies is given in Chapter 3.

In Chapter 4, ANNs are used to model the rainfall-runoff process using artificially-generated data, laboratory-experimental data and real, measured-catchment data to an exceptionally high degree of accuracy. The problem of extrapolation is described and a possible solution is posed for this problem. In Chapter 5, ANNs are applied to the problem of finding an accurate and stable solution of the pure advection equation. The resulting ANNs provide results that are at least as good as those obtained by more traditional numerical methods. Finally, in Chapter 6, ANNs are applied to the more general problem of data mining as exemplified by rainfall-runoff modelling, salt intrusion and sediment transportation. It is shown that the ANN outperforms more traditional, essentially manual, methods of data mining, and also provides significantly more accurate results than those obtained by another sub-symbolic paradigm, namely that of genetic programming.

Acknowledgements

In the 15 years since John Argue, of the then South Australian Institute of Technology, first actively encouraged me to continue my education overseas, I have enjoyed tremendous, unconditional support for the furtherment of my career and study from both IHE and the Danish Hydraulic Institute. I should like to express my sincere gratitude to both Wil Segeren and Asger Kej for the opportunities that they have made available to me.

I am indebted to all of my colleagues and friends at IHE and in Delft, who have continued to support and encourage me, sometimes through some very difficult times, when progress on this PhD had been reduced to no more than a slow, painful crawl.

However, none of this would have been possible without the continued patronage of my promoter, boss and, above all, friend, Michael B. Abbott, whose unwavering faith in my abilities and whose stimulating conversations have been an inspiration, not only to this work, but to my life in general

Finally, to my partner René, I wish to express my appreciation of his love and patience, and especially during the final stages of this, for me, monumental work.

1 Introduction

1.1 From Computational Hydraulics to Hydroinformatics

Already in the 1960s, Abbott (e.g. 1969) introduced the term 'computer hydraulics' to indicate that the traditional fields of hydraulics and hydraulic engineering would have to be reformulated to suit the possibilities and requirements of the discrete, sequential and recursive processes of digital computation. From rather disparate beginnings, *computational hydraulics* has established itself as an independent field of science in which the claims of hydraulic realism and data availability are balanced against the requirements and limitations of numerical stability and accuracy, and algorithmic simplicity (Abbott and Minns, 1997). In fact, the objective of the engineer in this area is to select a set of measurements and computations, out of the set of all possible measurements and all possible computations, that together will describe a process with the most relevance, reliability, and economy of means. The relevance, reliability and economy of the resulting model can, in turn, be related to the cost and the commercial value of the modelling service that it provides. It is possible to define the *utility* of a model according to the amount of information - which is proportional, following Szilard (1929), Shannon and Weaver (1949) and Brillouin (1956), to the amount of uncertainty that its presence resolves - and the degree to which it makes this information available (*see* Abbott, 1979). In classical Bernoullian terms we can relate any increment in information level, dI, to its associated increment in the utility of the model-processed information, $dU(I)$, by

$$dU(I) \propto \frac{dI}{I}$$

or

$$U(I) = Constant \times \ln\left(\frac{I}{I_0}\right)$$

where I_0 is a reference information state. We see then that any study aimed at increasing the utility of mathematical modelling is intrinsically linked to the amount and accessibility of the information, or resolution of uncertainty, that it makes available.

The reliance of computational hydraulics upon the digital computer has meant that this field of study has also been significantly influenced by the recent rapid technological advances in computer sciences, and in information technology in general. Similar developments can be identified in the fields of data acquisition, transmission, storage and retrieval. The computational engineer is now not only concerned with the selection and development of appropriate modelling tools but he must also deal with far greater quantities of information than ever before. This information must be made accessible to the modelling system, and then both raw and model-processed data must be made available to the client in a form that is understandable and

accessible, and often to a much wider audience than the client himself. The demands of this *information age* have led to a proliferation of electronic data-handling packages in the form of database management systems (DBMS), geographical information systems (GIS), supervisory control and data acquisition systems (SCADA), and many others.

The strengthening dominance of electronics among all enabling technologies is associated with an increasing and by now almost total dominance of digital representation. We speak of an electronic *encapsulation* of information and knowledge (Abbott, 1993). The act of encapsulating information and knowledge changes the very nature of the information and knowledge involved, as will be explained in the next section. Suffice to say that electronic encapsulation must also change the way in which an engineer accesses and uses the available information and knowledge. As will be explained later, there is a shift in paradigm away from a *model-based* approach to a more *data-based* approach.

This change of paradigm creates a new space in hydraulic engineering practice between the traditional model-based planning, design and decision-making methodologies of consulting engineers and the information collection and knowledge extraction techniques of hydraulics research. It is within this space that we see the emergence of *hydroinformatics*. Abbott (1994) speaks of a *Copernican revolution* in which the engineer is no longer asked to navigate his way in the world simply by building models of the world, but must increasingly learn to navigate through a world of models. The engineer now becomes less a personal carrier of all manner of detailed knowledge and information and more one who is adept at organising and integrating electronically-encapsulated knowledge and information.

Hydroinformatics may be described as the field of study that is concerned with the flow of information related to the flow of water - and all that it transports (Abbott, 1991). The hydroinformatics paradigm has emerged intuitively from the well-established field of computational hydraulics through the opportunities and obligations of modern information technology (IT). The results from standard, computer-based modelling systems merely form a 'carrier' or platform for the study of other aspects of the most immediate interest. The information necessary to describe and assess the state of any given body of water must also include a plethora of social, legal and environmental factors. The eventual physical, social and environmental impacts resulting from any action upon the water body can thereby be determined before the execution of the project. An important feature of a hydroinformatics system is that it allows the use of numerical simulations that are subject to constraints expressed in natural language (such as applicable legislation, contracts, agreements, etc.). Finally, this assessment process is enhanced by encapsulating expert knowledge and experience, merging this with measured data, and making this information available to hydro-scientists and engineers, such as in the form of computer-based, environmental impact assessment and decision support systems.

The very development of hydroinformatics and the corresponding value that its integrating function adds to each of its components separately, leads in its turn to an accelerated development of measuring equipment, to much more sophisticated SCADA systems, to new modelling capabilities, to new means to relate measurements and models through data assimilation, automatic constitutive equation generation, automatic calibration procedures and other such applications of inverse and adjoint methods, to new data base technologies, to new user interfaces and indeed to any number of other such developments.

Despite the new ground that has already been broken by hydroinformatics, these systems are far from fulfilling their potential. Hydroinformatics research does not remain limited to the fields of hydraulics and hydrology alone, but has recourse to the latest IT developments in the fields of artificial intelligence (including machine learning, evolutionary algorithms and artificial neural networks), artificial life, cellular or finite-state automata and other, previously unrelated sciences and technologies.

Through studying and exploiting elements of these, at first sight unrelated, sciences, hydroinformatics is producing new and innovative solutions to hydraulic and hydrological problems, as represented by real-time control and diagnosis, real-time forecasting, calibration of numerical models, data analysis and parameter estimation. (*see* Verwey *et al*, 1994; IAHR, 1994; Babović, 1996). Minns and Babović (1996) discuss this shift in paradigm as related to the field of hydrological modelling. In particular, these new approaches may be used to generate important components of physically-based, distributed hydrological modelling systems by *inducing* models or sub-models of individual physical processes based only upon measured data. These (sub)models may then replace whole systems of complex, non-linear, differential equations that would otherwise require great skills from the modeller to calibrate, and powerful computing devices to solve. In the separate, but not totally unrelated, field of ecological modelling, Babović and Barelta (1996) have investigated the application of these new modelling paradigms to modelling the higher trophic levels of aquatic ecosystems. They describe an individual-based modelling system in which the individual organism forms the logical base unit for the modelling of ecological phenomena. In artificial intelligence terminology we talk of *intelligent agents*. This hydroinformatics-type approach then integrates all of the individual sub-systems, or agents, and allows them to interact with their environment and neighbouring agents according to properties and characteristics that may vary between individuals. The overall population dynamics then *emerge* from the multitude of local interactions between the individuals.

1.2 Symbols, Signs and Models

In order to appreciate more fully the power of the new modelling paradigms, it is necessary to introduce a fundamental notion that expresses the essential difference between these new approaches and the more traditional modelling approaches. This is the notion of the differentiation between symbols and signs and thus between symbol manipulation and sign manipulation. Symbols are an artefact of our beliefs about the natural world. These symbols are tokens that *stand in the place* of the objects that they represent. A collection of tokens of this kind, however, does not constitute a model. It is only when we interpret a collection of tokens, thereby giving them 'meaning' or semantic content, so that each of the tokens in this collection *points towards* a certain natural phenomenon in the capacity of a *sign*, that we arrive at the level of a model. According to Heidegger (1927//1962, pp. 77,79,80):

> (...) We come across 'equipment' in *signs*. The word "sign" designates many kinds of things: not only may it stand for different *kinds* of signs, but Being-a-sign-for can itself be formalised as a *universal kind of relation*, so that the sign-structure itself provides an ontological clue for 'characterising' any entity whatsoever.

> But signs in the first instance, are themselves items of equipment whose specific character as equipment consists in *showing* [*zeigen*] or *indicating* [*anzeigen*].
>
> Signs (...) let what is ready-to-hand be encountered; more precisely, they let some context of it become accessible in such a way that our concernful dealings take on an orientation and hold it secure. A sign is not a Thing which stands to another Thing in the relationship of indicating; it is rather *an item of equipment which explicitly raises a totality of equipment into our circumspection so that together with it the worldly character of the ready-to-hand announces itself.* (...) Signs always indicate primarily 'wherein' one lives, where one's concern [*Besorgen*] dwells, what sort of involvement [*Bewandtnis*] there is with something.

The set of tokens that we recognise as the Navier Stokes equations for describing the motion of an incompressible fluid then constitute a model because each constituent equation constitutes 'a collection of signs that serves as a sign' that points to a belief that is so strongly experienced that we ascribe to it the status of a 'law of nature' (*see further* Abbott, 1992). There can only be a finite number of signs in this world created by the modeller and the potential infinity of details in the physical-world that cannot be described within this limited sign vocabulary are often gathered together in the form of assumptions and simplifications that have to be applied in order to read a meaning into the sequence of tokens that is the differential equation. If the modeller accepts the limitations imposed upon the model by the assumptions and simplifications, then he or she accepts that this sign vocabulary, or language, is the best available description of the physical processes being considered. In the traditional model-based paradigm, we input data, which itself is a collection of signs, into the model, that is formalised entirely in signs, and obtain output in the form of a collection of signs; which in this case serve as a sign to show or indicate the state of the system being modelled.

Of course we most commonly treat equations such as those of Navier Stokes as sequences of symbols that we can manipulate according to quite other laws than the laws of nature. In such cases we are not in the least interested in the 'physical meanings' of our manipulations: our symbols have for all purposes *replaced* our world of nature. Natural systems, however, rarely conform entirely to the assumptions imposed by the limited symbol vocabulary that we apply to the model. Subsequently, this symbolic language is commonly very restrictive for research into novel and innovative approaches to modelling. The inherent limitations of the traditional modelling approach and its associated language are exemplified in the *Philosophical Investigations* of Wittgenstein (Part I, §§ 2-3);

> Let us imagine a language ... (that is) ... meant to serve for communication between a builder A and an assistant B. A is building with building-stones: there are blocks, pillars, slabs and beams. B has to pass the stones, and that in the order in which A needs them. For this purpose they use a language consisting of the words 'block', 'pillar', 'slab' and 'beam'. A calls them out; - B brings the stone which he has *learnt* to bring at such-and-such a call. - Conceive this as a complete primitive language.
>
> [On the other hand,] Augustine, we might say, does describe a system of communication; only not everything that we call language is this system. And one has to say this in many cases where the question arises 'Is this an appropriate description or not?' The answer is:

> 'Yes, it is appropriate, *but only for this narrowly circumscribed region, not for the whole of what you were claiming to describe.*' (emphasis added)

In this simple example, the *universe of discourse* consists only of the words 'block', 'pillar', 'slab' and 'beam'. It would be impossible for the characters in this 'language game' ever to talk about 'doors', 'windows' or 'roofs' - let alone an entire house! Similarly, the hydraulic modeller is restricted in his or her description of an hydraulic system by the limited language of computational hydraulics. The description of this system can only be as detailed as the model that is to be used to simulate the physical and hydraulic processes, which in turn is restricted to the number of signs *cognizable* in the universe of discourse of the modeller. In the language game that is so circumscribed, no amount of extra measured data will ever change the basic structure of the underlying differential equations of the model, but may only be used to adjust certain calibration parameters in order to bring the results of model simulations closer to the observed and measured phenomena. Since functional similarity to the natural system is supposed to be comprehended by the equations themselves, it is the calibration parameters that must then capture the correspondence between the model and the real world. These parameters serve in effect as *error compensation devices* that artificially adjust the model results to compensate for the fundamental discrepancies that exist between the real world and its differential equation representation that underlies the model.

Calibration parameters are, however, usually not at all well-defined in nature. One may even ask 'What is the physical meaning of these parameters - how well are they *grounded*, and indeed are they grounded at all?'. We may indeed be able to read a certain 'physical meaning' into our calibration parameters, but they do not exist-as-such in our outer world of nature and are thus 'disconnected' in a fundamental way from the world which they are supposed to model. The rules that govern the way that the differential equations and the calibration parameters interact constitute one part of the grammar of the language of the hydraulics modeller. The traditional approach to modelling is one of simply manipulating and adjusting these collections of tokens (whether signs or symbols) in order to arrive at the best possible correspondence between model output and measured data. Nowadays we even recognise the process of symbol manipulation in the many commercial packages that claim to do just that (e.g. *Mathematica*, *Matlab*, etc.). Rarely is it possible for the modeller to create and incorporate new symbols and their associated signs into this language of discourse. The symbolic approach suffers from a rather thoroughly intractable problem of 'symbol grounding' (Harnad, 1990).

One of the greatest strengths of the new hydroinformatics approaches, which will be described in more detail later on, is their ability to identify relationships and to induce models based upon measured data without requiring a detailed preconceived knowledge of physical system characteristics *a priori*. One of the reasons for this is that many of these approaches manipulate the data at the level of the computer representation of the numbers. That is, the data are represented in our digital computers as *bits* and the operations upon this data then take place upon these individual bits. The modeller in this case has no direct influence upon the bits that convey the basic knowledge. After translating the underlying data that characterises our natural world into bit strings, the original symbols are then further irrelevant for the subsequent manipulations of the bits. The algorithm operates at the level of the bits, which now operate as

signs and no longer as symbols, and this way of working is referred to as a *sub-symbolic approach*.

The computer is entirely free in its manipulation of the bit strings; cutting them and rejoining them again at different places, flipping bits either randomly or in a controlled way. During this process the overall performance of the system can be observed and evaluated. This observation involves translating the bit-information back into data that then acts once more as a collection of signs that can be interpreted by the modeller. This resulting collection of signs is itself not necessarily a single sign as it may not only contain parameters that describe some physical phenomenon, but also signs expressed in natural language in the form of warnings or advice. The best solution of a problem is found when the interpreted output from the computer best matches some desired goals. The exploration of the search space to find the best solution takes place at the level of the bits within the computer - the sub-symbolic level - and , as such, is unrestricted by the limitations of the one or the other language of our symbolic world. The most important influence of the modeller in this process is then the translation or interpretation of the results being produced by the computer. These results should somehow 'make sense' to the modeller.

In his treatment of connectionist networks in cognitive modelling and artificial intelligence, Smolensky (1988, pp. 3,4) also highlights the fundamental differences between the symbolic and sub-symbolic paradigms. He explains how *language* provides the dominant theoretical model in cognitive science, so that formal cognitive models take their structure from the syntax of formal languages and their content from the semantics of natural language. In this way, cognitive descriptions of entities are constructed from symbols both in the semantic sense of referring to external objects and in the syntactical sense of being operated upon by symbol manipulation. The human brain is taken to be a machine for formal symbol manipulation. Several traditional artificial intelligence approaches to emulate this operation of the brain belong very obviously to the symbolic paradigm (*see also* Minns and Babović, 1996).

As pointed out already by Abbott (1991) however, such a 'formalist' view of the working of language has never been accepted at the higher levels of expression, where the *sounding* of the language and the associated coming-to-presence of its essential meaning have always been emphasised. Language, in this sense, can only be grounded in human experience generally: its working is existential.

In appearance, at least, the sub-symbolic paradigm incorporates cognitive descriptions constructed from entities that are merely *constituents* of the symbols used in the symbolic paradigm. Smolensky (1988, p.3) refers to these constituents as 'sub-symbols'. An entity represented by symbols in the symbolic paradigm can be represented by a much larger number of sub-symbols in the sub-symbolic paradigm. Operations in the symbolic paradigm usually consist of single discrete operations whereas in the sub-symbolic paradigm the result is obtained after a much larger number of 'fine-grained', numerical operations.

Within the above definitions it is then easy to see how the sub-symbolic methodologies have stimulated the shift from the earlier computational hydraulics paradigm, in which the model determines the way in which the data expresses itself, to the hydroinformatics paradigm, in which

the data is enabled to express its own choice of model. This enabling process describes another type of process that we may call *metamodelling*.

Within the computational hydraulics paradigm *the data is a collection of indicative signs that serves only to refine or orientate more precisely an already-explicated expressive sign*. Within the hydroinformatics paradigm, *the data serves to define also the expressive sign itself* - that which we usually identify as the nexus of 'the model' and which we commonly represent for our own purposes as a string of symbols, such as a set of equations. The role of the modeller is no longer that of choosing an appropriate model and subsequently fitting measured data to this model through the adjustment of calibration parameters. Rather, the modeller is dealing with the generation or evolution of a model in the form of an *information and knowledge encapsulator* that transforms the input data to some output signal through the intermediation of another kind of encapsulated knowledge. To put this in another way, we are now concerned to construct systems that themselves produce models: we are in effect aiming to construct 'metamodels'. The modeller must then select and prepare the appropriate data for input to the 'metamodel' and also provide an interpretation of the output from the 'metamodel', whereby it becomes a model 'for us'.

The selection of appropriate input data, the logical interpretation of the output and the methods for encapsulating the required kinds of knowledge into a sub-symbolic device may vary depending upon the choice of sub-symbolic paradigm. The applicability of some of these artificial-intelligence-based techniques, like evolutionary algorithms and artificial neural networks, together with their relative advantages and disadvantages in hydroinformatics will be discussed in more detail in chapter 2. The artificial neural network paradigm appears to be one of the most versatile and powerful of these sub-symbolic paradigms and it is in fact this particular paradigm that will be studied in greater detail throughout the rest of this work.

1.3 Electronic Knowledge Encapsulation

As mentioned earlier, in hydroinformatics we are generally dealing with various forms of knowledge and electronic knowledge encapsulators. For example, one form of knowledge is that of our beliefs concerning the physics, the chemistry, the biology, the economics and other forms of natural-scientific and social-economic knowledge that are encapsulated in numerical models. Similarly, knowledge expressed as facts and rules governing these facts are commonly encapsulated through the use of knowledge-based-system shells. The range and versatility of knowledge encapsulators that were already in general use in the hydroinformatics community in the early 1990's were described in some detail in a special edition of the *Journal of Hydraulic Research* (IAHR, 1994). It was clear already then that although the process of encapsulation of knowledge in traditional applications may be quite obvious to the practitioner who has just programmed the solution algorithm for a set of differential equations, or just entered a passage of text into a data field in a database, this process may be by no means so obvious when dealing with sub-symbolic devices. Can the information stored and manipulated within the sub-symbolic device be seen as knowledge? In order to answer this question we first consider the following definition (Barth, 1932//1975, p. 188)

> By the knowledge of an object by men we understand confirmation of their acquaintance with its reality in respect of its existence and its nature. But confirmation of their acquaintance means that the reality of the object concerned, its existence and nature, being true in themselves, now become in some way, and with some degree of clarity and distinctness, true for men too. Their acquaintance with it, instead of being a contingent and outward determination of their own existence, now becomes a necessary and inward determination. Knowing, they are affected by the object known. They no longer exist without it, but with it. In so far as they think of it, with the same confidence with which they dare to think in general they must think of it as a true reality, as true in its existence and nature. Whatever else and however else they may think of it, they must begin by thinking of the truth of its reality. Face to face with this truth they can no longer withdraw into themselves in order to affirm, question or deny it thence. Its truth has come home to them, has become their own. And in the process they themselves have become the truth's. This event, this confirmation, in contrast to mere cognizance, we call knowledge. Cognizance becomes knowledge when man becomes a responsible witness to its content.

We are confronted here also with the temporal nature of knowledge. In the modern-scientific sense, what we call knowledge, in so far as it refers to events at all, refers to *past* events (Park, 1978). In this data-based paradigm our original data sets have temporality; however, the encapsulated knowledge has, in principle, no temporality; it is *internalised* in the memory of the computer and is hence 'out of time'. It follows that the mental state that we call *perception* has temporality and that the state that occurs between perception and knowledge, that of *cognition*, represents the transition from the one state, of existing in time, to the other state, of existing out of time. Denbigh (1978) suggests that, in this way, memory in humans and higher animals may perhaps not significantly differ from computer memory. In the biological brain there are indications that short-term memories depend on short-term electrical excitations within the brain whilst long-term memories depend on changes in the molecular structure of the neurons, similar to the kind of imprinting that occurs in the memory bank of a computer.

Knowledge gives any collection of information its 'meaning' or semantic content. A traditional cognitive model may formalise domain knowledge in a linguistic structure such as "energy is conserved". An electronic knowledge encapsulator, on the other hand, takes our knowledge of our own material world and converts this knowledge to information in a truly sub-symbolic form, for example as a string of binary digits stored on the computer's hard disk drive. This information may then be processed back into a form that has meaning to the user and hence it provides the user with further knowledge. Whereas in §1.1 we spoke of the utility of a model only in terms of the amount and accessibility of information, we see here that there is a link between the accessibility of information and the knowledge that this information can provide us with. In fact, Abbott and Minns (1997, *see also* Abbott, 1993) in their description of the process:

$$\text{Knowledge} \rightarrow \text{Information} \rightarrow \text{Knowledge} \tag{1.3.1}$$

introduce the notion of the *social value* of the meaning content of the knowledge, V_S(«knowledge»), this being the value that society as a whole perceives this knowledge to be worth in material terms. They describe the inherent asymmetry in the process (1.3.1) in that the meaning content of the knowledge entering the left-hand-side of (1.3.1) is of a very different nature than the meaning content of the knowledge emerging on the right-hand side. For example,

the knowledge that is used in the construction of a model is of quite a different nature to the knowledge that the user of the model acquires through the use of the model. It is the associated increase in the social value, expressed as:

$$\Delta V_S = V_S(\text{«}\rightarrow \text{knowledge»}) - V_S(\text{«knowledge} \rightarrow \text{»}) \qquad (1.3.2)$$

that justifies the use of electronic knowledge encapsulators.

Smolensky (1988, p.4) refers to science as a 'cultural activity' and gives three extremely valuable properties of formalised *cultural knowledge*, namely:

- *public access*: it is of limited social value to have knowledge that resides purely in one individual,
- *reliability*: it is of questionable social value to have knowledge formulated in such a way that different users draw different conclusions from it,
- *formality and universality*: for cultural propagation of knowledge it is helpful if novices with little or no experience with a task can be given a means for performing that task, and thereby a means for acquiring experience.

Thus, at a cultural level, the goal of any knowledge encapsulator is to express knowledge in a form that can be accessed and used reliably by different people, even inexperienced people. In the past, linguistic formulations of knowledge have usually proven to be the most suitable for this purpose.

With an electronic knowledge encapsulator, the user only interacts externally with the encapsulating device in the acts of supplying the collection of signs represented by 'knowledge →', and subsequently interpreting the collection of signs represented by '→ knowledge'. The fact that knowledge and signs are thereby so inextricably related is rather eloquently described by Foucault (1966//1970, p.59):

> (...) From the seventeenth century onward, the whole domain of the sign is divided between the certain and the probable: that is to say, there can no longer be an unknown sign, a mute mark. This is not because men are in possession of all the possible signs, but because there can be no sign until there exists a *known* possibility of substitution between two *known* elements. The sign does not wait in silence for the coming of a man capable of recognising it: it can be constituted only by an act of knowing.
>
> (...) From now on, (...), it is within knowledge itself that the sign is to perform its signifying function; it is from knowledge that it will borrow its certainty or probability.

Furthermore we can identify two levels of knowledge in the encapsulation process. As Abbott (1993) states, this process of electronic encapsulation leads necessarily to a restructuring of the knowledge itself, and the form of our knowledge plays an important role in the choice of knowledge encapsulator. For a description of these forms we shall return to Foucault (1966//1970, p.72):

> What makes the totality of the Classical *episteme* possible is primarily the relation to a knowledge of order. When dealing with the ordering of simple natures, one has recourse to a mathesis, of which the universal method is algebra. When dealing with the order of

> complex natures (representations in general, as they are given in experience), one has to constitute a *taxinomia*, and to do that one has to establish a system of signs. These signs are to the order of composite natures what algebra is to the order of simple natures. But in so far as empirical representations must be analysable into simple natures, it is clear that the *taxinomia* relates wholly to the mathesis; on the other hand, since the perception of proofs is only one particular case of representation in general, one can equally well say that mathesis is only one particular case of *taxinomia*. Similarly the signs established by thought itself constitute, as it were, an algebra of complex representations; and algebra, inversely, is a method of providing simple natures with signs and of operating upon those signs.

These various forms of knowledge and their relationships to each other through signs is shown in Fig. 1.1.

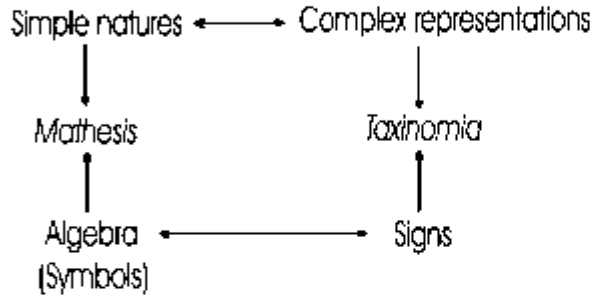


Fig. 1.1 General science of order (adapted from Foucault, 1966//1970, p.72)

Abbott (1993) uses the simplest example of the solution of a set of differential equations using a numerical scheme to illustrate the *mathesis* of computational hydraulics. A set of equations has its own *mathesis* that allows us to recognise this set of statements as a model of some hydraulic phenomenon, in as much as it is a collection of signs that serve as a sign. The solution scheme has again its own *mathesis* that allows us to translate the set of equation statements, written in a standard language of the continuum, into a set of statements in what is, in effect, a first-order language, which is 'understandable' to our programmable machine.

On the other hand, if we consider an example from ecological modelling, encapsulation of knowledge at the level of the *mathesis* in this case would only provide us with the simplest paradigm in modelling population dynamics, such as the prey-predator equations of Lotka-Volterra (*see* Maynard-Smith, 1973):

$$\begin{aligned} dN/dt &= aN - bN^2 - \alpha NP \\ dP/dt &= -cP + \beta NP \end{aligned} \qquad (1.3.3)$$

in which each and every organism in the prey population, N, and in the predator population, P, is considered only in relation to the total quantity of its functional type. The effect of the predators upon the prey population is only measured by the 'functional response' term, αNP. Babović and Baretta (1996) argue that the multiplicity of details in a complex ecological system

can however best be modelled by an individual-based approach and the individuals within each functional type appear to be most conveniently represented by intelligent agents. The *taxinomia* of the intelligent agent approach encapsulates all of the knowledge about the *behaviour of the individuals*. This knowledge is composed of both declarative ('knowing-that') and procedural ('knowing-how') types of knowledge which each individual is presumed to possess (Minns and Babović, 1991).This may not only include physics-based equations of mass and energy balances, but also intelligence and biological aspects of competition, cooperation, breeding and such principles as 'the survival of the fittest'.

Finally, we can see that *mathesis* refers only to rules (or even 'laws') governing the flows of information, usually expressed in the form of symbols. Once more, Foucault (1966//1970, p. 74) points out that , in the strict sense, *mathesis* is a science of equalities, and therefore of attributions and judgements; it is the science of *truth*. *Taxinomia* , on the other hand, is the science of classifications and orderings, dealing essentially with identities and differences; it is the knowledge of *beings*. We might also express this by saying that whereas *taxinomia* betokens the laws of nature, *mathesis* betokens 'the laws of the laws of nature'.

As will be described in more detail in Chapter 2, a sub-symbolic device, like an artificial neural network, can encapsulate knowledge at the level of the *taxinomia* by establishing a relation between some input, that may be a collection of any signs represented by 'knowledge ', and some output, that is another collection of signs represented by ' knowledge'. This relation is established during a *learning* phase in the device, which has the aim of reducing the *visible differences* between the output from the device and some *desired* or *expected* results, which is yet another collection of signs that are usually determined by measurements or from our own detailed knowledge of the system being described. The mathematical expression of the visible difference between these collections of signs may be in the form of a simple least-squares error between sets of real numbers, or may include system state descriptions expressed in natural language. The power of the sub-symbolic approach is described most clearly by Smolensky (*in* Rumelhart *et al*, 1986, Vol. 1, p. 261):

> Nowhere is the contrast between the symbolic and subsymbolic approaches to cognition more dramatic than in learning. Learning a new concept in the symbolic approach entails creating something like a new schema. Because schemata are such large and complex knowledge structures, developing automatic procedures for generating them in original and flexible ways is extremely difficult.
>
> In the subsymbolic account, by contrast, a new schema comes into being gradually, as the strengths of atoms slowly shifts in response to environmental observation, and new groups of coherent atoms slowly gain important influence in the processing. During learning, there need never be any decision that "now is the time to create and store a new schema". Or rather, if such a decision is made, it is by the modeller *observing* the evolving cognitive system and not by the system itself.

Babović (1996) suggests that the promotion of emergent computation as a paradigmatic approach to modelling advocates an alternative and more natural way of thinking. He describes a number of sub-symbolic paradigms that are based upon the concepts of Darwinian evolution; the so-called *evolutionary algorithms*. Computational systems of entities or agents that are allowed to

interact and genetically evolve form implicit global patterns at the macroscopic level. Thus solutions emerge whose signs point to other objects in the natural world that could not have been indicated by the collection of symbols contained in a preconceived model.

Finally, some sub-symbolic devices, like genetic programming, may even be used to encapsulate knowledge in the sense of *mathesis* by producing what appear to be physically realistic mathematical formulae of a physical phenomenon (*see* Babovic and Abbott, 1997). The relative advantages and disadvantages of this approach as compared to artificial neural networks will be discussed later, in Chapter 6.

1.4 Thesis Outline

This work is composed of seven chapters. A short overview of the material to be presented in the following chapters is given here.

Chapter 2 describes the history and developments of computations involving artificial neurons or perceptrons. A simple example is used to demonstrate the learning ability and the linear classification properties of a single perceptron. After a short discussion of the universal computation abilities of the perceptron as well as some of its shortcomings, there follows a rather detailed description of the multi-layer perceptron and the generalised delta rule most commonly used to train multi-layer neural networks. This chapter concludes with a brief description of some other common artificial neural networks.

Due to the importance of the sub-symbolic paradigm to the whole problem of knowledge encapsulation and machine learning, Chapter 3 provides a general description of some other sub-symbolic techniques that are also enjoying some success in certain hydroinformatics applications. In particular, the performance of one of these paradigms, that of genetic programming, is compared to the performance of artificial neural networks in Chapter 6.

Chapter 4 describes the results of the application of artificial neural networks to the modelling of the rainfall-runoff process. Both real and experimental data are used to demonstrate the ability of the ANN to learn a usable relationship between the rainfall and the runoff, based only upon measured rainfall depths and discharges from the catchment. The problem of the extrapolation of results using an ANN is addressed and a possible solution to this problem for these types of hydrological applications is presented.

Chapter 5 is concerned with the problems where the exact mathematical formulation of the relationship is already well known, e.g. in the form of a differential equation, but where accurate solutions of the mathematical formulation are difficult to achieve. The scalar wave equation is used as an example of this type of problem. Both linear and non-linear ANNs are applied to find solutions to the pure advection problem.

Chapter 6 describes the application of ANNs to the most general problem of data mining. IN these problems, a relationship is being sought amongst vast quantities of raw data for which the existence of some deterministic relationship between certain variables has not yet been accurately

established. The results of the ANNs are compared with those obtained by applying genetic programming to the same raw data. All of these results are then compared to those obtained from more traditional, manual methods.

Chapter 7 summarises the conclusions drawn from this present work and highlights the strengths and weaknesses of artificial neural networks when applied to hydroinformatics problems. It further identifies some related application areas which should be investigated further in the future.

The appendix provides a general description of the problems of overfitting and generalisation as they may be encountered with any artificial neural network and describes a procedure that can help avoid these problems.

2 Artificial Neural Networks

2.1 General

The sub-symbolic paradigms that appear to offer the greatest potential in hydroinformatics are those of artificial neural networks, evolutionary algorithms and cellular automata. This chapter includes a detailed description of the range and applications of artificial neural networks, which is the paradigm that forms the central theme of this study. The subsequent chapter contains a short overview of the methodologies of some evolutionary algorithms and cellular automata approaches in order to allow comparisons of the relative advantages and disadvantages of these methodologies as compared with those of artificial neural networks.

2.2 Biological Inspiration of the Artificial Neuron

The ability of the brain to perform difficult operations and to recognise complex patterns, even if these patterns are distorted by 'noise', has formed the subject matter of the discipline of cognitive psychology that has in turn strongly influenced the study of artificial intelligence (AI).The particular ability of the brain to learn from experience without a predefined knowledge of underlying physical relationships makes it an exceptionally flexible and powerful calculating device that AI researchers have long tried to mimic.

At the same time, other researchers have been devoted to reproducing, or modelling, physical phenomena by making use of electronic computational machines to solve ever-increasingly complex continuum (ordinary and partial differential, and integral) equations and related empirical relationships. These researchers are supported by a rapid increase in the computational capacity of modern computers and an emerging recognition of the advantages of massively parallel computation (parallel distributed processing) that allows the required calculations to proceed with ever-increasing speed. However, although the design and construction of the hardware for parallel computation is relatively straightforward, the software currently available for creating algorithms to utilise this parallel architecture for solving partial differential and other such equations efficiently is still quite limited.

These two groups of researchers, pursuing what appear to be quite different goals, have found a common ground in the field of artificial neural networks. ANNs are not an exact computational representation of the human brain but are merely *inspired* by our (limited) understanding of the operations that take place within the brain. The human brain consists of approximately 10 to 100 billion (10^9) brain cells or *neurons*. Each of these neurons is connected to approximately 10,000 other neurons. The time taken by a single biological neuron to respond to a given stimulus is about 1 millisecond. If we consider that a complete task can be solved by the brain in about 0.5

seconds then, by taking into account some delays caused by the travel time of information between the neurons, we see that the brain performs about 100 processing steps during the execution of the task. Now, the time for a single switching operation in a digital computer is of the order of magnitude of 1 nanosecond (10^{-9} s), which is about 1 million times faster than the response time of a biological neuron. The von Neuman architecture of our modern computers restricts us to a sequential execution of switching operations or calculations. The brain, on the other hand, compensates for the relatively slow switching speed by making use of a massively parallel configuration. The inspiration for artificial neural networks then lies in the desire to emulate the functionality of the human brain on a conventional computer in which the lack of parallelism is compensated by the sheer speed of computation.

One of the earliest breakthroughs in neural computation was inspired by the rather misguided apprehension that it should be possible to emulate the cognitive functionality of the brain simply by reproducing the physical mechanisms of brain activity. At the simplest level, it is possible to schematise a biological neuron as shown in Fig. 2.1.

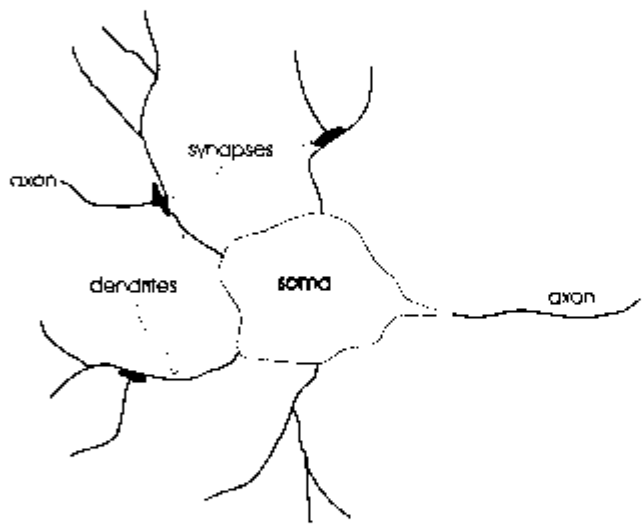


Fig. 2.1 Schematisation of a biological neuron

The main cell body of a biological neuron is called the *soma*. Attached to the soma is a tree-like network of nerve fibres called *dendrites*. The dendrites carry signals *to* the soma in the form of electrical pulses that have originated from other neurons. The soma processes the incoming information and may produce an output signal in the *axon* in the form of another electrical pulse or *action potential*. The axon ramifies into various branches that make *synapses* onto the dendrites and somas of other neurons. The axon is a sort of non-linear threshold device that is 'fired' when the potential within the soma rises above a certain threshold level. Chemicals called *neurotransmitters* are released from the synapse when the potential of the synapse has been sufficiently raised by the action potential in the axon. The effect of the neurotransmitters on the receiving dendrite or soma is to raise (excite) or lower (inhibit) the electrical potential inside the body of the receiving cell. For a more complete description on how biological neurons actually perform computations reference is made to the work of Hopfield (1994).

An extremely over- simplified description of the operation of biological neurons forms the basis of the model neuron proposed by McCulloch and Pitts (1943). The McCulloch-Pitts neuron, schematised in Fig. 2.2, consists of a simple summation device attached to a binary threshold unit.

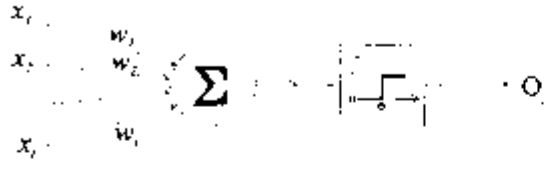


Fig. 2.2 Schematic diagram of the McCulloch-Pitts model neuron

The model neuron computes the weighted sum of incoming signals x_i and produces an output of one or zero depending upon whether this sum is above or below a given threshold value, θ. This can be expressed mathematically as:

$$O_j = H\left(\sum_i w_i x_i - \theta_j\right) \tag{2.2.1}$$

where $H(\xi)$ is the *Heaviside step function*:

$$H(\xi) = \begin{cases} 0 & \text{if } \xi \leq 0 \\ 1 & \text{otherwise} \end{cases} \tag{2.2.2}$$

The weight w_i represents the excitory or inhibitory effect of the synapse attached to connection i. McCulloch and Pitts (1943) showed how a synchronous assembly of these model neurons could compute any logical function for a suitable selection of the weights w_i. This demonstrated the ability of these devices to compute numerically also. In fact, systems of model neurons provide a complete computational model capable, in principle, of performing the same computations that can be performed by any digital computer.

The ability of these artificial neurons to compute, however, is only a first step towards emulating the functionality of the human brain. One of the most fundamental properties of the human brain is its ability to *learn* from examples. For the simple model neuron this means that we require a mechanism for choosing the connection weights so that we can calculate something useful. In practice we 'teach' the system to perform a desired computation by iterative adjustments of the w_i. To continue the biological analogy, we note that learning in the human brain often involves reinforcing behaviour that we wish to see repeated and discouraging incorrect or 'bad' behaviour. Hebb (1949) theorised that learning occurred in brains through the modification of synapses, and that repeated firings across a synapse increase its sensitivity and hence the future likelihood of its firing. A particular positive stimulus then causes a strong association between a certain group of cells. For a network of model neurons, Hebb proposed that a reasonable and biologically plausible mechanism would be to strengthen the connections between elements of the network only when both the presynaptic and postsynaptic units were active simultaneously.

Rumelhart *et al* (1986, Vol I, p. 36) give the following rule for adjusting connection strengths:

> *Adjust the strength of the connection between (two) units A and B in proportion to the product of their simultaneous activation.*

This natural extension of Hebb's original formulation allows both positive and negative activation values resulting in positive (excitory) or negative (inhibitory) changes to the connection strengths. This rule is called the *Hebb rule* and this learning paradigm is referred to as *Hebbian learning*.

2.3 Perceptrons and Linear Classifiers

Model neurons, connected up in a simple fashion were given the name 'perceptrons' by Rosenblatt in 1962. Minsky and Papert (1969/1988) provided a rather thorough mathematical treatment of perceptrons and their ability to *learn* to distinguish between classes of patterns. Following these earlier works, we define a perceptron as a device that is capable of computing any *predicate* $\psi(X)$ of a given pattern X. The pattern X is described by the set of features $\varphi_i(X)$ and we define the predicate:

$$\psi(X) = 1 \text{ if and only if } \Sigma w_i\varphi_i \geq \theta \qquad (2.3.1)$$

where w_i is a set of numbers or *weights* and θ is some *threshold value*. Defined in this way, $\psi(X)$ is said to be a *linear threshold function* and is sketched in Fig. 2.3.

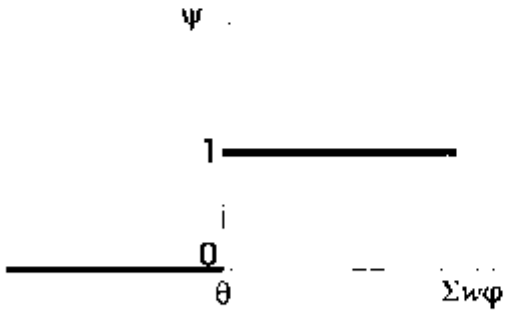


Fig. 2.3 Linear threshold function for the predicate $\psi(X)$

That is, given a set of φ_i, this can be assigned to one of two possible classes, defined by the two possible states of the predicate ψ, by simply setting the coefficients w_i and the threshold value θ to suitable values. Minsky and Papert (1969/1988, pp. 164-170) present a procedure for adjusting these values in a so-called *learning process* until the correct values are obtained. In this procedure, we represent the set of features φ_i of any given pattern X in vector notation as Φ. The set of weights w_i can then be represented in vector notation as $\mathbf{W}$. We now replace $\Sigma w_i\varphi_i$ by the scalar product $\mathbf{W} \cdot \Phi$. If we assume that our input patterns come from a space that has two classes, F′ and F″, we can define a predicate:

$\psi = 1$ if and only if $\mathbf{W} \cdot \Phi > 0$ (2.3.2)

which corresponds to $X \in F^+$. The predicate (2.3.2) implicitly identifies the second class of pattern, F^-, in that $\mathbf{W} \cdot \Phi < 0$ corresponds to $X \in F^-$.

If we select a set of weights $\mathbf{W}$ for our perceptron and then present it with a pattern X, it will calculate the sum $\mathbf{W} \cdot \Phi$. If this sum is positive and $X \in F^+$ then the answer is correct. However, if we now take a different pattern $X \in F^+$ and calculate a sum that is negative, then this answer is wrong and the weights in the perceptron should be adjusted so that the resulting sum becomes positive. That is, the values of the weight coefficients should be increased. The question then arises of determining by how much the weight values should in all events be increased. Clearly, if any of the φ_i is zero then these values and their corresponding weight coefficients cannot be blamed for the incorrect total of the summation. Correspondingly, there is no reason to make any changes in the values of these particular weights. The most elegant way of increasing only those coefficients that correspond to non-zero values of φ_i is simply to add the vector Φ to the vector $\mathbf{W}$. In a similar way, if the sum $\mathbf{W} \cdot \Phi$ had been positive for a pattern $X \in F^-$ then the corresponding adjustment to the weights would simply involve subtracting the vector Φ from the vector $\mathbf{W}$. This procedure is summarised in Fig. 2.4.

START:	choose any value for $\mathbf{W}$
TEST:	choose an X if $X \in F^+$ and $\mathbf{W} \cdot \Phi > 0$ goto TEST if $X \in F^+$ and $\mathbf{W} \cdot \Phi \leq 0$ goto ADD if $X \in F^-$ and $\mathbf{W} \cdot \Phi < 0$ goto TEST if $X \in F^-$ and $\mathbf{W} \cdot \Phi \geq 0$ goto SUBTRACT
ADD:	replace $\mathbf{W}$ with $\mathbf{W} + \Phi$ goto TEST
SUBTRACT:	replace $\mathbf{W}$ with $\mathbf{W} - \Phi$ goto TEST.

Fig 2.4 Algorithm for the classification of two linearly-separable classes

A variation on the above procedure was proposed by Widrow and Hoff (1960) based upon the principles of *gradient descent* in which the changes to the weights should be proportional to the gradient of the error function at that location. If we define an error or cost function for a given pattern $\lambda = (x_1, x_2, \ldots x_i)$ as:

$$E_\lambda = \frac{1}{2}\left(d_\lambda - \sum_i w_i x_i \right)^2 \qquad (2.3.3)$$

where d_λ is the desired output for pattern λ, i.e. $d_\lambda = 0$ if $\lambda \in F^-$, or $d_\lambda = 1$ if $\lambda \in F^+$.

Then we can calculate the gradient of the error function as:

$$\frac{\partial E_\lambda}{\partial w_i} = -(d_\lambda - \sum_i w_i x_i) x_i \qquad (2.3.4)$$

Note that during the learning process the output is not passed through the step function, although actual classification is effected by using the step function to produce d_λ. This is of course equivalent to saying that $\Sigma\, w_i x_i = 1$ is sufficient to classify $\lambda \in F^+$, and $\Sigma\, w_i x_i = 0$ is sufficient to classify $\lambda \in F^-$. Adjusting the weights by an amount proportional to (2.3.4) then gives:

$$\Delta w_{i,\lambda} = \eta \frac{\partial E_\lambda}{\partial w_i} = \eta (d_\lambda - \sum_i w_i x_i) x_i = \eta \delta_\lambda x_i \qquad (2.3.5)$$

where η is a multiplication factor between 0 and 1 that has the effect of slowing down the change in the weights, thus forcing the network to take smaller steps towards the solution. Eq. (2.3.5) is referred to as the *Widrow-Hoff delta rule*.

The basic algorithm in Fig. 2.4 is so simple and clearly defined that Rosenblatt (1962) proved the existence of a *perceptron convergence theorem* that defines conditions under which this procedure is guaranteed to find a correct set of values. The perceptron convergence theorem states that whatever choice is made in START and whatever choice function is used in TEST, the vector **W** will be changed only a finite number of times. In other words, if the sets F^+ and F^- are linearly separable then the above program will find a solution vector **W** for which the predicate (2.3.2) will separate them. The mathematical proof of this is given in Minsky and Papert (1969/1988, pp. 168-170, *see also* Beale and Jackson, 1990, pp. 53-57).

A simple example will suffice to demonstrate the convergence of the above procedure to a desired solution. Consider the four patterns divided into two classes as given in Table 2.1. Each of the patterns can be identified by the properties (φ_1, φ_2), and have been assigned either to class F^+ or F^-. The patterns are also shown on Fig. 2.5, where the patterns in class F^+ are indicated by a '+' symbol and the patterns in class F^- are indicated by a 'O' symbol.

Table 2.1 Example patterns and associated classes

Pattern	φ_1	φ_2	Class
1	2	3	F^+
2	3	6	F^-
3	4	1	F^+
4	5	4	F^+

A weight vector, $\mathbf{W}_3$, is also drawn on the same figure (dashed line). Generally speaking, the vector space of the weight vector, $\mathbf{W} = (w_1, w_2)$ would not be the same vector space as that of the pattern vectors, $\Phi = (\varphi_1, \varphi_2)$. Even though the scalar product $\mathbf{W} \cdot \Phi$ implies that $\mathbf{W}$ is a linear operator on the patterns themselves, it in fact operates on spaces of functional operators on the patterns. For convenience of demonstration we have used the same vector space for both the patterns and the weights.

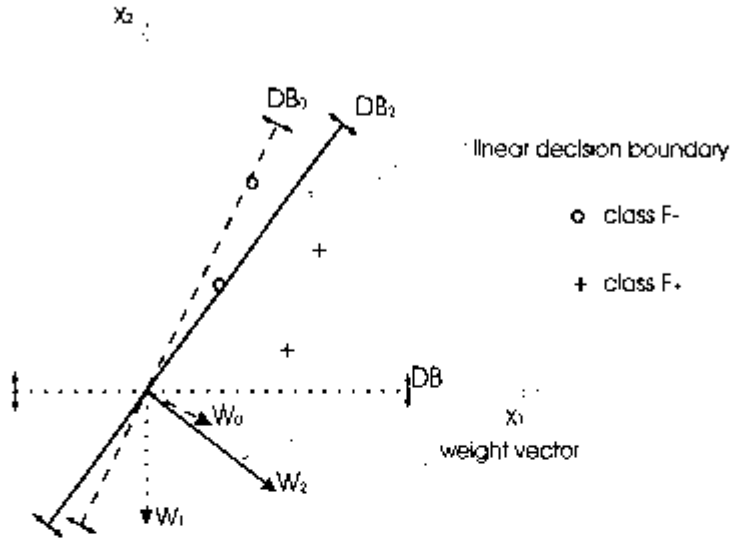


Fig. 2.5 Progression of the weight vector and its associated linear decision boundary during the classification of two classes (see Table 2.2)

Fig. 2.5 shows the progression of the weight vector, $\mathbf{W}$, and its associated linear decision boundary, **DB**, during the classification process, starting from the initial 'guess' of $\mathbf{W}_0$ (dashed line) and ending up with the solution $\mathbf{W}_1$ (thick solid line). The calculations for each step of the above procedure are given in Table 2.2.

Table 2.2 Calculations accompanying the classification example in Fig. 2.3

iteration	weight vector, $\mathbf{W}$	pattern vector, Φ	class	$\mathbf{W} \cdot \Phi$	action
0	$\mathbf{W}_0 = (2,-1)$	$\Phi_1 = (2,3)$	F	2x2 1x3 = 1	$\mathbf{W} = \mathbf{W} - \Phi$
1	$\mathbf{W}_1$ (0,-4)	$\Phi_2 = (3,6)$	F	0x3 - 4x6 = -24	no change to $\mathbf{W}$
2	$\mathbf{W}_2 = (0,-4)$	$\Phi_3 = (4,1)$	F	0x4 - 4x1 = -4	$\mathbf{W} = \mathbf{W} + \Phi$
3	$\mathbf{W}_3 = (4,-3)$	$\Phi_4 = (5,4)$	F	4x5 - 3x4 = 8	no change to $\mathbf{W}$
4	$\mathbf{W}_3 = (4,-3)$	$\Phi_1 = (2,3)$	F	4x2 - 3x3 -1	no change to $\mathbf{W}$
5	$\mathbf{W}_3 = (4,-3)$	$\Phi_2 = (3,6)$	F	4x3 3x6 = -6	no change to $\mathbf{W}$
6	$\mathbf{W}_3 = (4,-3)$	$\Phi_3 = (4,1)$	F	4x4 - 3x1 = 13	no more changes to $\mathbf{W}$

From Fig. 2.5, we see that the decision boundary is a straight line drawn perpendicular to the weight vector. This orientation of the decision boundary is obvious when we consider the expansion of the scalar product:

$$\mathbf{W} \cdot \Phi = |\mathbf{W}| \, |\Phi| \cos \gamma \tag{2.3.6}$$

where γ is the angle between the vector $\mathbf{W}$ and the vector Φ. It is clear to see that the scalar product (2.3.6) will indeed be positive for all values of γ between +90 and −90 degrees. This then implies a straight decision-boundary line where all patterns located 'above' this line will result in a positive value of (2.3.6) and hence belong to class F^+, and all patterns located 'below' this line will result in a negative value of (2.3.6) and hence belong to class F^-.

Linear classifiers like the one just described can be used to separate more than two classes by arranging many decision boundaries and performing several tests to satisfy the conditions for each class (*see* Beale and Jackson, 1990, p. 31). For example, in a problem consisting of classes A, B or C it would be possible to establish a decision boundary that would separate A from BC; and then, if the answer were not A, to establish a decision boundary to separate B from C. Similarly, for difficult class-boundary conditions, the decision surface can be split up in a piecewise fashion as shown in Fig. 2.6.

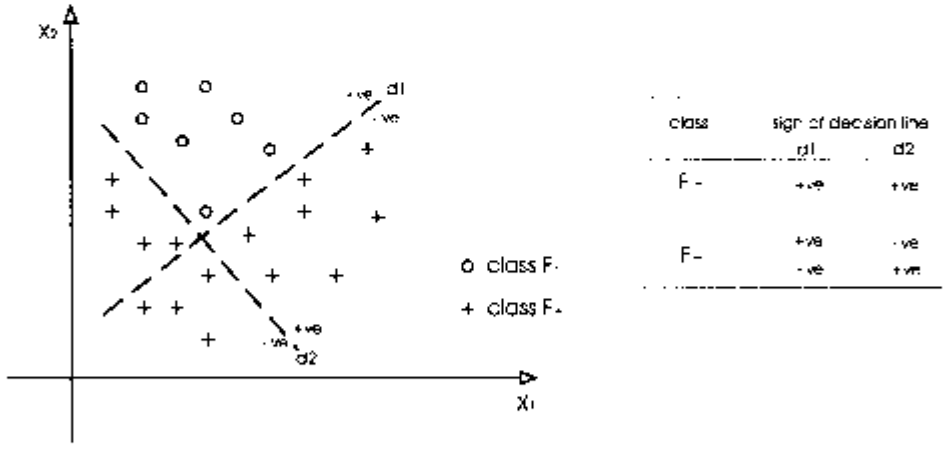


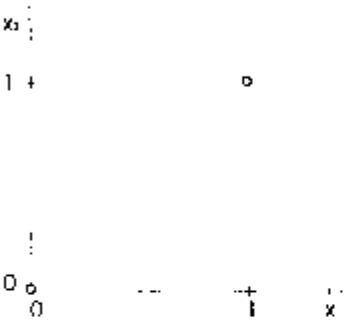
Fig. 2.6 Piecewise linear classification for a non-linearly separable pattern (adapted from Beale and Jackson, 1990, p. 31)

In summary, therefore, we see that Rosenblatt's claim that a perceptron would learn anything that it was possible to program it to do refers only to the problem of classification of inputs that are linearly separable. This limitation was recognised by Minsky and Papert (1969/1988) in which they quite extensively 'proved' the inability of a perceptron of finite order to recognise connectedness in any given figure. This was most clearly stated by the authors (*ibid.*, p. 113):

> It is our conviction that the deterioration of the perceptron's ability to recognise patterns embedded in other contexts is a serious deterrent to using it in real, practical situations. Of

course this deficiency can be mitigated by embedding the perceptron in a more serial process - one in which the figure of interest is isolated and separated from its context in an earlier phase. But this presupposes enough recognition ability, in the 'pre-processing' phase, to discern and remove the most commonly encountered contextual disturbances, and this may be much harder than the 'processing' phase.

Furthermore, the perceptron is quite incapable of classifying patterns in many situations where the patterns are *linearly inseparable*. The simplest example of the exclusive-or (XOR) logic function shows that for the case with just two inputs and one output, the XOR function table, as given in Fig. 2.7, produces a pattern space of zeros and ones in which it is impossible to draw a single straight line to separate the two classes.



XOR function table

x_1	x_2	XOR	class
0	0	0	o
1	0	1	+
0	1	1	+
1	1	0	o

Fig. 2.7 The linearly inseparable XOR problem in pattern space

Minsky and Papert recognised that some of the above mentioned limitations could be circumvented by adding another layer of logic to the machine to permit 'and-ing' two perceptrons together. However, they expressed serious doubts about the possibility of discovering a universal theorem, similar to the perceptron convergence theorem for individual perceptrons, that could be generally applied to these multi-layered machines. In particular (*ibid.*, pp. 231-232):

> The problem of extension is not merely technical. It is also strategic. The perceptron has shown itself worthy of study despite (and even because of!) its severe limitations. It has many features to attract attention: its linearity; its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension is sterile. Perhaps some powerful convergence theorem will be discovered, or some profound reason for the failure to produce an interesting 'learning theorem' for the multi-layered machine will be found.

2.4 Multi-layer Networks

The publication of Minsky and Papert's rather damning book in 1969 seemed to bring developments in this field to a halt. Very little progress was made in addressing Minsky and

Papert's criticisms until 1986 when Rumelhart, McClelland and the PDP Research Group reignited interest in multi-layer perceptrons by publishing details of a successful learning algorithm for automatically adjusting the weights in all layers of the network.

The following simple example shows how a multi-layered perceptron system can in fact represent the XOR problem. Following the technique described in Fig. 2.6 we see that it is possible to construct a perceptron that can detect only the input pattern (0,1) and a different perceptron to detect only the input pattern (1,0). This is schematised in Fig. 2.8.

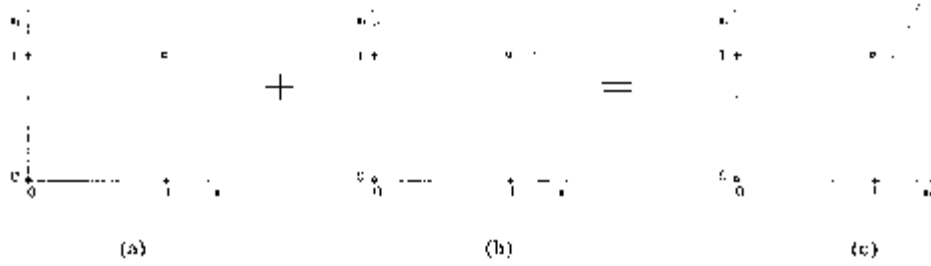
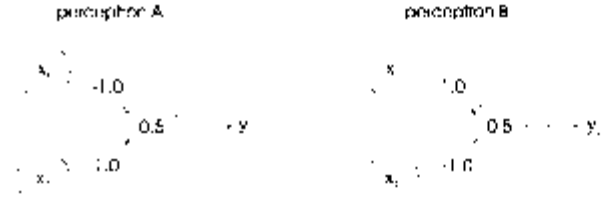


Fig 2.8 Combining perceptrons to solve the XOR problem

Using the predicate (2.3.1) we arrive at two individual perceptrons for each classification task as shown in Fig. 2.9 (a) and (b) respectively. The threshold value θ of the step function in each case is shown inside the circle representing the summation and threshold device and the weights of the connections are shown alongside the connections. We see that perceptron A of Fig. 2.8a only produces an output of 1 for the input pattern (0,1) and that the perceptron B of Fig. 2.8b only produces an output of 1 for the input pattern (1,0).



Input patterns	Output patterns
(0,0)	0
(0,1)	1
(1,0)	0
(1,1)	0

(a)

Input patterns	Output patterns
(0,0)	0
(0,1)	0
(1,0)	1
(1,1)	0

(b)

Fig 2.9 (a) Perceptron A to detect only input pattern (0,1) and (b) perceptron B to detect only input pattern (1,0)

We now take the outputs of perceptrons A and B to be the inputs to a new perceptron C as shown in Fig. 2.10.

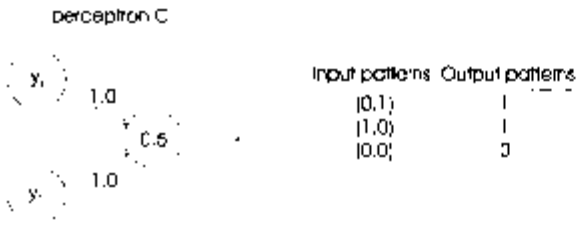


Fig 2.10 Perceptron C that takes its input only from perceptrons A and B

Combinations of the output signals from A and B now provide only three (but unique) input patterns to perceptron C. These three patterns and their correspondence to the original input patterns are given in Table 2.3. The resulting weights and threshold value for this new perceptron are also shown in Fig. 2.10.

Table 2.3 Correspondence of input patterns to perceptron C with the original patterns of the XOR problem

Input patterns to perceptron C	Original input patterns
(0,1)	(1,0)
(1,0)	(0,1)
(0,0)	(0,0) and (1,1)

This entire network is schematised in Fig. 2.11. This network is referred to as a *multi-layer perceptron* (MLP). We often call this artefact an *artificial neural network* (ANN) to stress that it is by no means the same as its *biological* inspiration. The MLP consists of an extra layer of units between the input and the output layer. These units are not connected directly to the outside world as are the units in the input and output layers. For this reason, these units are referred to as *hidden units* and the entire layer as a *hidden layer*.

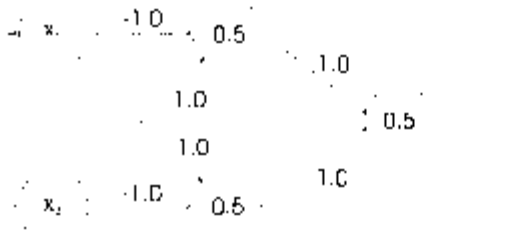


Fig 2.11 A multi-layer perceptron network for the XOR problem

The units in the hidden layer play an important role in the *internal representation* of the input patterns. If there were no hidden layer, the network would only be able to map similar input patterns to similar output patterns in which the similarity of the patterns is determined only by their overlap. Whenever the representation provided by the outside world is such that the similarity structure of the input and output patterns are very different, a network without internal representations will be unable to perform the necessary mappings. In the above example, we see

that the hidden layer of the XOR network in Fig. 2.11 provides a generalisation of the four input patterns into three internal representations as shown in Table 2.3. The network is able to generalise the input signal in which both bits are 'on', i.e. (1,1), and the input signal in which both bits are 'off', i.e. (0,0) into a single internal representation of (0,0) even though the original input patterns appear to have the least degree of overlap.

The ability of artificial neural networks to generalise is one of the main reasons for their widespread popularity. Not only can they generalise pattern features on discrete pattern sets, such as in the XOR problem, but if the input and output data have some deterministic relationship then the ANN can often induce this relationship and interpolate on the input and output data in a sensible way. The major problem then remains of establishing a method for selecting and adjusting the weights of the ANN connections. The weights and the threshold values in the ANN shown in Fig 2.11 have all been fixed manually. It is in fact impossible for this particular network to *learn* the XOR relationship on its own. Since Hebbian learning corresponds to strengthening the connections between active input and output units, it is impossible to strengthen the correct parts of the network since the actual inputs are effectively masked-off from the output units by the intermediate layer.

Furthermore, the linear threshold function that is used here (Fig. 2.3) gives us no information about the magnitude of the weighted sum of inputs, $\Sigma w_i\varphi_i$ that turns the output either 'on', i.e. $\Sigma w_i\varphi_i > \theta$, or 'off', i.e. $\Sigma w_i\varphi_i < \theta$. Weights that only just turn the neuron on or off should be adjusted at a different rate than weights that cause a significant overshoot or undershoot of the threshold value. One way around this problem is to introduce a different threshold function that has similar properties to the linear threshold function in the extremities but gives us more information about the proximity of the value of the weighted sum to the threshold value. The most commonly used function for this purpose is the *sigmoid* or *logistic function*:

$$y = \frac{1}{1 + e^{-(\Sigma w_i\varphi_i - \theta)}} \tag{2.4.1}$$

which is graphed in Fig. 2.12.

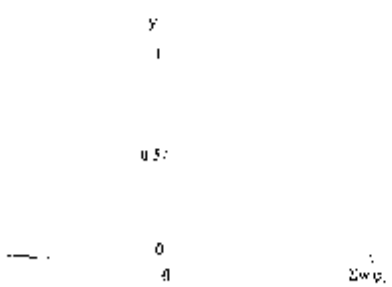


Fig. 2.12 The sigmoid or logistic threshold function

Note that the effect of the threshold value θ in (2.4.1) is to *offset* the centre of the sigmoid function from zero. The same effect could also be obtained by introducing an extra input in the

perceptron, say $i = 0$, for which $\varphi_0 = -1$ and $w_{0j} = \theta$. The equation describing the sigmoid function can then be simply written as:

$$y = \frac{1}{1 + e^{-\sum w_i \varphi_i}} \tag{2.4.2}$$

where the lower limit of the summation is now $i = 0$ instead of $i = 1$.

2.4.1 Generalised delta rule

Rumelhart *et al* (1986) published a new learning rule that could also adjust the weights in all of the intermediate layers of a any general multi-layer perceptron. This is a generalisation of the Widrow-Hoff delta rule (2.3.5) and is subsequently referred to as the *generalised delta rule*. In this case, the MLP must utilise a nonlinear activation function like (2.4.2) and the network must be arranged in layers where the units in each layer receive input only from units in the previous layer and the output from each unit goes only to units in the following layer. There is no interconnection of units within one layer and no direct connections of units across intermediate layers. A schematisation of this general MLP and its nomenclature are given in Fig. 2.13.

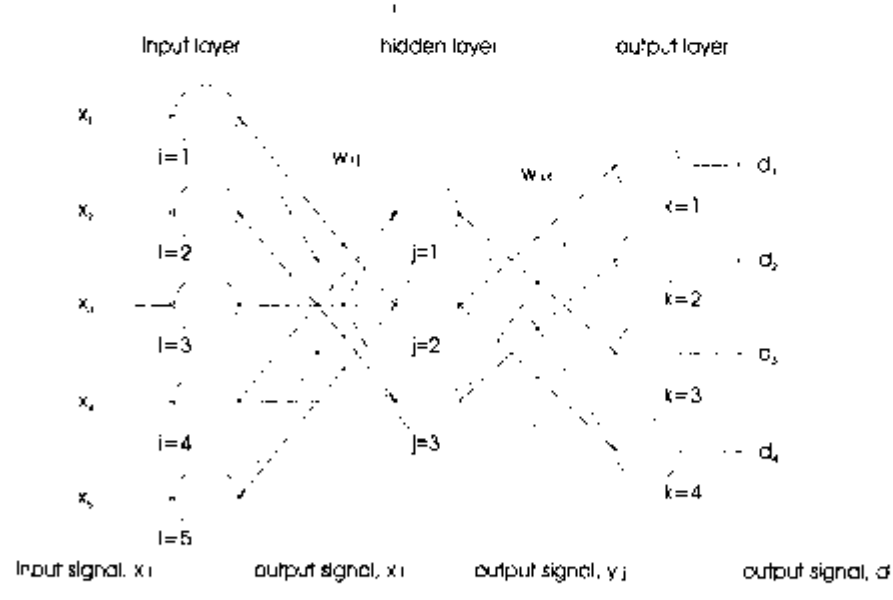


Fig 2.13 A general multi-layer perceptron (MLP) used for the derivation of the generalised delta rule

Note that the input layer in Fig. 2.13 does not perform any operations upon the input signal but simply sends the x_i to the units in the hidden layer. The output from a unit, j in the hidden layer for a given training pattern, λ is given by:

$$y_{j,\lambda} = f(\sum_{i=0} w_{i,j}\ x_{i,\lambda}) \tag{2.4.3}$$

where the activation function f is of the form (2.4.2).

The output from a unit, k in the output layer is then:

$$d_{k,\lambda} = f(\sum_{j=0} w_{j,k}\ y_{j,\lambda}) \tag{2.4.4}$$

Rumelhart *et al* (1986, pp. 323 *et seq.*) introduce the general expression:

$$o_{j,\lambda} = f(net_{j,\lambda}) \tag{2.4.5}$$

where

$$net_{j,\lambda} = \sum_{i} w_{i,j}\ o_{i,\lambda} \tag{2.4.6}$$

Similarly for the output layer:

$$o_{k,\lambda} = f(net_{k,\lambda}) \tag{2.4.7}$$

where:

$$net_{k,\lambda} = \sum_{j} w_{j,k}\ o_{j,\lambda} \tag{2.4.8}$$

The error function is now defined as:

$$E_{\lambda} = \frac{1}{2}\sum_{k} (d_{k,\lambda} - o_{k,\lambda})^2 \tag{2.4.9}$$

We then write

$$\frac{\partial E_{\lambda}}{\partial w_{i,j}} = \frac{\partial E_{\lambda}}{\partial net_{j,\lambda}} \frac{\partial net_{j,\lambda}}{\partial w_{i,j}} \tag{2.4.10}$$

From (2.4.8) we can write:

$$\frac{\partial net_{j,\lambda}}{\partial w_{i,j}} = \frac{\partial}{\partial w_{i,j}} \sum_{i} w_{i,j}\ o_{i,\lambda} = o_{i,\lambda} \tag{2.4.11}$$

If we define:

$$\delta_{j,\lambda} = -\frac{\partial E_{\lambda}}{\partial net_{j,\lambda}} \tag{2.4.12}$$

and then substitute (2.4.11) and (2.4.12) into (2.4.10), we get:

$$\frac{\partial E_\lambda}{\partial w_{i,j}} = -\delta_{j,\lambda}\, o_{i,\lambda} \tag{2.4.13}$$

By applying the principles of gradient descent, we must then change our weights in proportion to the amount given by (2.4.13), i.e.:

$$\left(\Delta w_{i,j}\right)_\lambda = \eta\, \delta_{j,\lambda}\, o_{i,\lambda} \tag{2.4.14}$$

which is very similar to the Widrow-Hoff delta rule (2.3.5).

To calculate $\delta_{j,\lambda}$ for each unit we write (2.4.12) as:

$$\delta_{j,\lambda} = -\frac{\partial E_\lambda}{\partial net_{j,\lambda}} = -\frac{\partial E_\lambda}{\partial o_{j,\lambda}} \frac{\partial o_{j,\lambda}}{\partial net_{j,\lambda}} \tag{2.4.15}$$

From (2.4.5) we have:

$$\frac{\partial o_{j,\lambda}}{\partial net_{j,\lambda}} = f'(net_{j,\lambda}) \tag{2.4.16}$$

and then we can write:

$$\begin{aligned}
\frac{\partial E_\lambda}{\partial o_{j,\lambda}} &= \sum_k \frac{\partial E_\lambda}{\partial net_{k,\lambda}} \frac{\partial net_{k,\lambda}}{\partial o_{j,\lambda}} \\
&= \sum_k \frac{\partial E_\lambda}{\partial net_{k,\lambda}} \frac{\partial}{\partial o_{j,\lambda}} \sum_i w_{i,k}\, o_{i,\lambda} \\
&= \sum_k \frac{\partial E_\lambda}{\partial net_{k,\lambda}}\, w_{j,k} \\
&= -\sum_k \delta_{k,\lambda}\, w_{j,k}
\end{aligned} \tag{2.4.17}$$

Thus, substituting (2.4.16) and (2.4.17) into (2.4.15), we get:

$$\delta_{j,\lambda} = f'(net_{j,\lambda}) \sum_k \delta_{k,\lambda}\, w_{j,k} \tag{2.4.18}$$

Similarly, for the output units:

$$\delta_{k,\lambda} = -\frac{\partial E_\lambda}{\partial net_{k,\lambda}} = -\frac{\partial E_\lambda}{\partial o_{k,\lambda}} \frac{\partial o_{k,\lambda}}{\partial net_{k,\lambda}} \tag{2.4.19}$$

where it follows from our definition of E_λ in (2.4.9) that:

$$\frac{\partial E_\lambda}{\partial o_{k,\lambda}} = -(d_{k,\lambda} - o_{k,\lambda}) \tag{2.4.20}$$

so that (2.4.19) becomes:

$$\delta_{k,\lambda} = -(d_{k,\lambda} - o_{k,\lambda})\, f'(net_{k,\lambda}) \tag{2.4.21}$$

and

$$(\Delta w_{j,k})_\lambda = \eta\, \delta_{k,\lambda}\, o_{j,\lambda} \tag{2.4.22}$$

The application of the generalised delta rule thus involves two phases. During the first phase, the input, x_λ, is presented to the network and propagated forwards through the network to the output layer. Here the desired output, $d_{k,\lambda}$, and the computed output, $o_{k,\lambda}$, are compared to each other and the error signal $\delta_{k,\lambda}$ is computed from (2.4.21) and the corresponding weight adjustments from (2.4.22). In the second phase, this error signal is propagated *backwards* through the network to each intermediate layer. This method of propagating the error signal backwards through the network leads to the generalised delta rule being often referred to as *error back-propagation*. At each intermediate layer the error signal $\delta_{j,\lambda}$ is computed from (2.4.18) and the associated weight adjustments from (2.4.14). This process is repeated for every layer until the input layer is reached.

The next input/output tuple is then applied and the connection weights readjusted to minimise this new error. This procedure is repeated until all training data sets have been applied. The whole process is then repeated again, starting from the first data set once more and continuing until the total error for all data sets is sufficiently small and subsequent adjustments to the weights are inconsequential. The ANN is now said to have *learned* a relationship between the input and output training data sets, The exact form of this relationship cannot be extracted from the ANN but rather is encapsulated in the stored series of weights and connections between nodes. The absolute values of the individual weights cannot normally be interpreted to have any deeper physical meaning.

Although the error back-propagation method does not guarantee convergence to an optimal solution since local minima may exist, it appears in practice that it leads to solutions in almost every case (Rumelhart *et al*, 1994). In fact, standard multi-layer, feed-forward networks, with only one hidden layer have been found capable of approximating any measurable function to any desired degree of accuracy (Hornik *et al*, 1989). Errors in representation would appear to arise only from having insufficient hidden units or the relationships themselves being insufficiently deterministic. MLPs are indeed *universal approximators*.

In the above derivations the derivative of the activation function $f'(net)$ plays an important role. For the generalised delta rule to succeed, we need an activation function for which the derivative exists. It is obvious that the discontinuous, linear threshold function used earlier (Fig. 2.3) cannot be used here as its derivative is infinite at the threshold value and zero elsewhere. The sigmoid function, on the other hand, is extremely well suited to this method due to its very simple derivative. Substituting (2.4.2) into (2.4.5) gives:

$$o_{j,\lambda} = f(net_{j,\lambda}) = \frac{1}{1 + e^{-net_{j,\lambda}}} \tag{2.4.23}$$

so that the derivative can be derived as:

$$f'(net_{j,\lambda}) = \frac{e^{net_{j,\lambda}}}{\left(1 + e^{-net_{j,\lambda}}\right)^2}$$

$$= \frac{1}{1 + e^{-net_{j,\lambda}}} \left[1 - \frac{1}{1 + e^{-net_{j,\lambda}}}\right]$$

$$= f(net_{j,\lambda}) \left[1 - f(net_{j,\lambda})\right]$$

which can be written as:

$$f'(net_{j,\lambda}) = o_{j,\lambda} (1 - o_{j,\lambda}) \tag{2.4.24}$$

The derivative is therefore a simple function of the outputs.

Many computer packages are now available that implement the generalised delta rule to train MLPs. In some cases, other activation functions as well as the sigmoid function have been implemented which may affect the performance of the MLP or its rate of learning. These other activation functions may include, for example:

- linear function; $f(net) = net$ (2.4.25)
- hyperbolic tangent function; $f(net) = \tanh(net)$ (2.4.26)

Finally, the rate of learning may also be improved by modifying the generalised delta rule to include a *momentum term*. In this case the expression for updating the weights becomes:

$$\Delta w_{i,j}(n) = \eta \, \delta_{j,\lambda} \, o_{i,\lambda} + \alpha \, \Delta w_{i,j}(n-1) \tag{2.4.27}$$

where $\Delta w(n)$ refers to the changes to be made to the weights for the current iteration, n and $\Delta w(n-1)$ refers to the weight change in the previous iteration. The momentum term, α, determines the effect of past weight changes on the current direction of movement in the weight space. The introduction of such a momentum term is of great assistance in speeding up convergence along shallow gradients by allowing the approach to the solution to 'pick-up speed' in the downhill direction. Furthermore, it is useful in weight spaces containing long ravines that are characterised by both sharp curvatures across the ravine and a gently sloping floor. The sharp curvature tends to cause oscillations across the ravine. To prevent this it is necessary to take very small steps, but this slows down the progress considerably along the ravine. The momentum term allows the net

to filter out the high curvature and thus it allows the effective change in weights to be so much larger..

Multilayer perceptrons that are trained using error back-propagation appear at the moment to be the most common artificial neural network used in engineering practice. For this reason, the present work restricts itself primarily to the use of MLPs. The computer software used for almost all of the simulations presented in this study were performed using the WinNN software developed by Dr. Y. Danon.

2.5 Some Other Common Types of Artificial Neural Network

Although the standard MLP is the most popular type of artificial neural network, it is perhaps useful here to mention some other common forms of ANN for purposes of comparison.

2.5.1 Radial basis function networks

Radial basis function (RBF) networks provide a number of techniques that essentially preprocess the data and transform it into a higher dimensional space in which the classes are linearly separable. Whereas the standard MLP builds its classifications on the basis of *hyperplanes* defined by the weighted sums $\Sigma_i w_{ji} x_i$ that are arguments to non-linear functions, the RBF approach uses *hyper-ellipses* to partition the pattern space (Beale and Jackson, 1990, p. 94 *et seq*).

In an RBF network the hidden layer units (Fig. 2.13) do not contain sigmoid functions, but are replaced by response functions dependent upon the positions of the data relative to some centre point, i.e.:

$$\phi(\|x - w\|) \tag{2.5.1}$$

where $\|...\|$ denotes some distance measure of x from a centre point w. The distance measure is most often chosen to be Euclidean, so that:

$$\delta_j = \|x - w\|_2 = \sqrt{\sum_i (x_i - w_{ji})^2} \tag{2.5.2}$$

and the most commonly used response function is a Gaussian function, i.e.:

$$\phi(\delta_j) = y_j = e^{-\lambda \delta_j^2} \tag{2.5.3}$$

where λ is some constant.

Thus, unit j gives a maximum response to input vectors near w_j. Each hidden unit therefore occupies a region in the input space centred upon w_j that Hertz *et al* (1991) refer to as its *receptive field*. The idea is then to pave the input space with these receptive fields. If an input vector x lies in the middle of the receptive field for unit j then only unit j will be activated. If the input vector lies between two receptive field centres, then the network will make a smooth interpolation between the two units.

The output from the RBF network is a linear combination of the basis function (2.5.3). This approach is guaranteed to produce a function that fits all data points as long as there is a basis function for each input to be classified. Having one hidden unit for each input, however, means that noisy or anomalous data points will also be classified, which will give problems of generalisation for new 'unseen' data points. The generalisation properties of the RBF network can be improved by reducing the number of hidden units. The selection of the coefficients for the linear combination of basis function outputs is then simply a problem of linear optimisation, which is guaranteed to find a globally optimal solution. The problem of training an RBF network then is that of selecting sufficient hidden units to get an acceptable fit to the data. This is normally done using a trial and error procedure.

Some practical results of RBF networks applied to the problem of rainfall-runoff modelling are given in Mason *et al* (1996) and a comparison of RBF networks with MLPs is given in Dibike (1997). These results demonstrated that the training time of RBF networks was usually significantly less than for equivalent MLP networks. Furthermore, the RBF networks provided a superior performance over MLPs when dealing with only small numbers of input data sets. However, the generalisation properties of the RBF networks deteriorated significantly as the number of input data sets increased and were subsequently out-performed by the MLP networks in this case.

2.5.2 Hopfield networks

Hopfield (1982, *see also* Hertz *et al*, 1991, pp. 11-41) presented an *autoassociative* network consisting of perceptrons that are very similar to those used in the standard MLP. The problem of associative memory is to store a set of n patterns x_i in such a way that when presented with a new pattern z_i the network responds by producing whichever one of the stored patterns most closely resembles z_i. Some practical applications of associative memory are in the recognition and reconstruction of images, or in the retrieval of bibliographic information from partial references.

The Hopfield net is a fully-connected network in which all of the units are connected to all other units, as schematised in Fig. 2.14. The weight distribution is also symmetric, such that the weight of the connection from node i to node j is the same as the weight of the connection from node j to node i, i.e. $w_{ij} = w_{ji}$. Each unit in the Hopfield net acts in a similar way to the single layer perceptron, in that it has as its input the weighted sum $\Sigma\, w_{ij}\, x_j$ and it produces an output by passing this weighted sum through a linear threshold function, as shown earlier in Fig. 2.3.

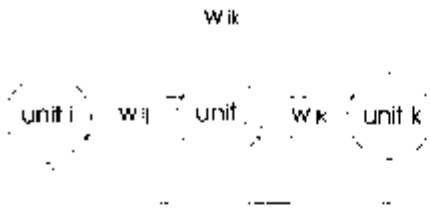


Fig 2.14 The Hopfield network

The Hopfield network has no specific input or output units. The input to the network consists of a vector of binary (0,1) or bipolar (1,+1) values that are applied to all of the units simultaneously. The network is then left alone as it cycles through a succession of states until it converges on a stable solution. The output of the network is taken to be the values of each of the units when the network has reached a stable, steady state.

Training of the network involves setting the weights of the connections based upon a set of n training patterns using the simple rule:

$$w_{ij} = \frac{1}{N} \sum_{\lambda=1}^{n} x_i^{\lambda} x_j^{\lambda}$$

where N is the number of units in the network. It can be shown (Hertz *et al*, 1991, pp. 35 *et seq*) that a Hopfield net can accurately store and recall a maximum of $0.138N$ patterns.

A similar network to the Hopfield net is the so-called *Boltzmann machine*, which has a similar structure and learning algorithm to the Hopfield net but is coupled with a probabilistic update rule. The Boltzmann machine uses simulated annealing in order to help it to converge to global minima (*see also* Beale and Jackson, 1990, pp. 133-163).

2.5.3 Kohonen networks

One of the most important features of the artificial neural networks mentioned so far is that the learning algorithms are based on Hebbian learning. That is, training takes place by introducing some input data at one end of the network and the 'desired' output data is presented at the other end. The ANN is 'told' that a relationship exists between this input and output data and that it should find this representation. This process known as *supervised learning*.

An alternative to the Hebbian, supervised learning paradigm is an unsupervised learning paradigm based on *competitive learning* (Rumelhart *et al*, 1986, Vol. 1, pp. 151 *et seq*). The basic components of a competitive learning scheme are:

- start with a set of units that are the same except for some randomly distributed parameter which makes each of them respond slightly differently to a set of input patterns;
- limit the 'strength' of each unit;
- allow the units to compete in some way for the right to respond to a given subset of inputs.

The net result of applying this learning paradigm is that individual units learn to specialise on sets of similar patterns and thus become *feature detectors* or *pattern classifiers*. Similar inputs should be classified as being in the same category and so should 'fire' the same output unit.

Kohonen's *self-organising map* is a network that uses unsupervised learning to model features found in the training data. The Kohonen net consists of a single layer of neurons. Each neuron in this layer is connected to all input units. As in the MLP, the input units themselves perform no computations but simply distribute the input signal to the layer of neurons. Fig. 2.15 shows a two-dimensional Kohonen network, although it is also possible to arrange the neurons in a single row and thus form a one-dimensional network.

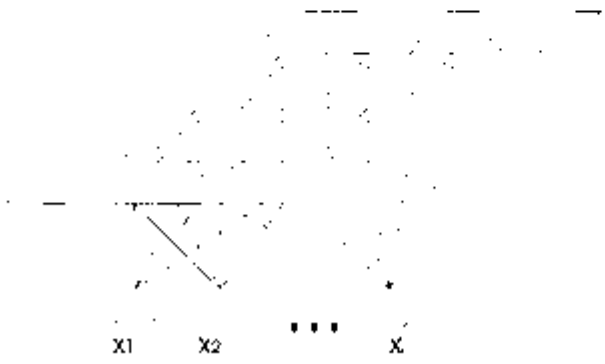


Fig. 2.15 A Kohonen feature map (adapted from Beale and Jackson, 1990, p. 110)

The learning process organises the neurons on the grid into local neighbourhoods that act as feature classifiers on the input data. There is no 'desired' output data to tell the network which patterns should be assigned to which classifications. The training procedure involves firstly initialising the weights of the connections between the input units and the neurons in the upper layer. Each input pattern x_i is then introduced to the network and the network must 'decide' which neuron is associated with this particular pattern. The neuron to be fired is usually determined by calculating some *similarity measure* D_k for each neuron, k, and assigning the 'winner' to be the neuron with the largest value of D_k. A very commonly used similarity measure is the scalar product of the weight vector of the neuron with the input vector, i.e.:

$$D_{k,\lambda} = \sum_{i} w_{k,i} x_{i,\lambda}$$

The similarity of the winning neuron to the input is then increased by increasing the corresponding weight connections to this neuron.

The location of the winning neuron in the one- or two-dimensional arrangement of output neurons may also convey some information about the input pattern. Similar input patterns should fire neurons that are nearby in the output space. This suggests the construction of a topology preserving map from the space of possible inputs to the line or plane of output units. This *topographic feature map* is essentially a mapping that preserves neighbourhood relations. Neighbourhood relations are maintained in the Kohonen net by updating the weights not only to

the winning neuron but also to the neurons in its immediate neighbourhood. The number of immediate neighbours to be updated depends upon a neighbourhood function that decreases in size as the training time progresses, thus localising the area of maximum activity.

After training the network, the Kohonen map must be interpreted to determine which classification has actually taken place. This analysis often takes the form of a visual inspection of the output from the network and the weight connections. Shawkat Ali (1997) describes some common techniques that include:

- *output activity map*; the output from a single input vector is plotted on a one- or two-dimensional map showing the actual value of the similarity measure at each unit.

- *vector position* or *cluster map*; the positions of the winning neuron for each input vector are displayed on the output map. Separate clusters are formed by the ensemble of winning units.

- *count map*; this map is formed by simply counting the number of hits for each output neuron for the entire input data set.

- *topological feature map*; when a stable weight configuration has been achieved after training, Kohonen networks act as an associative memory device of the input space. If the input space is two- or three-dimensional it is possible to study the topological structure of the input space by plotting the weight vectors graphically.

Shawkat Ali (1997) demonstrated the ability of Kohonen nets to identify 'meaningful' clusters within very large data sets. The number of clusters identified is significantly less than the total number of data sets. This data reduction property of the Kohonen net means that it becomes possible to represent high-order, multi-dimensional data by certain lower-dimensional features. This property of Kohonen nets means that it may be possible to use these nets to 'pre-process' large quantities of raw data into smaller data sets containing only 'significant features'. This reduced data set could then be used to train a traditional, feed-forward, multi-layer neural network in order to discover the relationship between these significant features. This could significantly reduce the training time of the ANN compared to the situation in which all of the unprocessed, raw data were used.

3 Some Other Sub-symbolic Paradigms

3.1 Introduction

The power of the sub-symbolic paradigm does not restrict itself to the study of artificial neural networks. Abbott and Minns (1997, pp. 459-463) highlight the advantages of a sub-symbolic approach to modelling in their discussion of the computational hydraulics of turbulence. A subject like computational hydraulics quickly ties itself up in knots of formidable complexity - and the fact that we tie these knots in the first place is a direct consequence of our use of specific languages. Each such language works with sets of tokens that are almost exclusively sets of symbols with definite physical representations. At some level, our problems of complexity then become grounded in our use of definite *physical symbol systems* of a certain and, in practice, ever more extended complexity. Such systems have been studied very extensively over the millennia, but in recent times they have attracted a special attention in computer science, and then especially in those branches that pass under the rubrics of artificial intelligence and computer science (e.g. Newell, 1980).

A careful study of such systems, even when these are generalised to the greatest extent possible, points up certain limitations that are inherent in their use. The basic conclusion is that the use of physical symbol systems, although essential to human existence itself, necessarily introduces constraints, and that in many circumstances these constraints are unacceptable (Abbott, 1991, pp. 94-106). There is thus a major incentive to 'escape' from symbolic paradigms as far as possible, and instead to resort as far as possible to sub-symbolic paradigms. In these paradigms the tokens are signs and not symbols, so that they no longer, 'stand in front of ' the indicated object but instead 'point towards' that object. We ourselves set up rules about how these signs are to interact, but we do not ourselves realise the interaction itself - any more than we can foresee the results of this interaction. In effect, we deliberately relinquish our semantic preconceptions concerning the tokens. There is then of course an apparent contradiction, in that we still write codes using symbols even as we claim that these codes themselves 'operate at the sub-symbolic level'.

The three main divisions within the sub-symbolic paradigm are those of artificial neural networks, evolutionary algorithms and cellular automata. For the sake of comparison with the neural networks described in the previous chapter, a brief description of the latter two areas is given below.

3.2 Evolutionary Algorithms

Minns and Babović (1996) provide a succinct comparison between artificial neural networks and evolutionary algorithms when applied to problems of hydrological modelling. Evolutionary algorithms (EAs) are computer-based simulation engines that mimic in a grossly simplified way the evolutionary processes occurring in nature. The fundamental idea behind EAs is indeed that of plagiarising those particular processes postulated by Darwin (1859) in his seminal work on the theory of evolution of species. Darwinian theory depicts the adaptation of species to its environment as one of *natural selection*. Perceived in this way, all species currently inhabiting our planet (and for that matter, all species that have ever lived on this planet) are actually the result of this process of adaptation.

The EA approach to problem solving is one in which solutions to the problem are evolved rather than the problems being solved directly. The family of evolutionary algorithms may be characterised by four main streams: Evolution Strategies (Schwefel, 1981), Evolutionary Programming (Fogel *et al*, 1966), Genetic Algorithms (Holland, 1975) and Genetic Programming (Koza, 1992).

Although different and applied for quite different purposes, all EAs share a common conceptual base. In principle, an initial population of individuals is created in a computer and allowed to evolve using the principles of *inheritance* (so that offspring resemble parents), *variability* (the process of offspring creation is not perfect - some mutations occur) and *selection* (more fit, or 'better', individuals are allowed to reproduce more often and less fit individuals less often so that the 'genealogical' trees of the latter will 'die out' with time).

Fig. 3.1 depicts the main processes that make up an evolutionary algorithm. From an initial, typically randomly generated, population of individuals the fittest entities are selected to be altered by genetic operators exemplified by *crossover* (corresponding to sexual reproduction) and *mutation*. Selection is performed on the basis of a certain fitness criterion in which the fitter individuals are selected more often. Crossover combines two genotypes by exchanging sub-strings around a randomly selected point. Mutation simply flips a randomly selected bit.

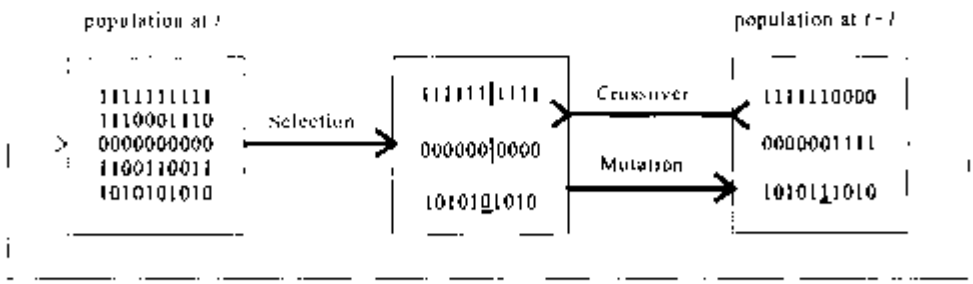


Fig. 3.1 Schematic illustration of an evolutionary algorithm

Similar to the processes of nature, one should distinguish between the evolving entity's *genotype* and its *phenotype*. The genotype is essentially a code to be executed (such as a code in the DNA strand in humans), and the phenotype represents the result of the execution of this code (such as a living person). The information exchange between evolving entities (parents) occurs at the level

of the genotypes; however, it is the phenotypes in which we are really interested. The phenotype is in effect an interpretation of a genotype in a problem domain. This interpretation can take the form of any feasible mapping. One of the main advantages of EAs is their domain independence. EAs can evolve almost anything, given an appropriate representation of the evolving structures. For example, for optimisation and constraint satisfaction purposes, genotypes are typically interpreted as independent variables of a function to be optimised. Several applications of genetic algorithms (GAs) that make use of this kind of mapping and with specific emphasis on water resources are described by Babović (1993).

In so-called *learning classifier systems (LCS)*, as introduced by Holland (1986), phenotypes take the appearance of rules in evolving knowledge-bases. LCSs are actually built on the top of ordinary GAs, and continuously augment the knowledge-base with new and better-performing rules, thus avoiding a rigid and static tree structure. LCSs thus open avenues towards automatic model enhancement through the process of machine learning (*see* Wilson, 1994).

In genetic programming (GP), the evolutionary force is directed towards the creation of models that take a symbolic form. In this evolutionary paradigm, evolving entities are presented with a collection of data, and the evolutionary process is expected to result in a closed-form symbolic expression that describes the data. In principle, GP evolves tree structures representing symbolic expressions in Reverse Polish Notation. The nodes in this tree structure are user-defined. This means that they can be algebraic operators, such as *sin*, *log*, +, -, etc., or can take a form of *if-then-else* rules, making use of logical operators such as *OR*, *AND*, etc. (*see* Walker *et al*, 1993, *and* Babović and Abbott, 1997).

It is extremely difficult ,if not impossible, to describe the full potential of EAs and their applications in such a limited space. Here, however, two essential properties of EAs should be highlighted:

- Evolutionary Algorithms are sub-symbolic models of computation. As was suggested before, the exchange of information between evolving entities occurs at the level of the genotypes. The phenotypes represent or contain the *meaning* encoded in the genotypes. This meaning (or semantic interpretation) is acquired through both a *mapping function* (from genotype to phenotype) and an interaction of the phenotype with its environment. This applies for the entire EA family. GP in its most rudimentary form can be understood as a method for evolving trees which acquire meaning only when they are confronted with the problem domain;
- The most important phenomenon in relation to EA performance is that it attains its knowledge about its environment through interaction with this environment. The knowledge about a problem that is being solved does not exist explicitly within the EA-based problem-solver before the problem-solving (*i.e.* evolutionary) process is initiated. This knowledge is acquired through the process of survival of the fittest. The consequence of this is that the process of solving problems actually transforms to one of adequately describing the problem and then letting the solution to the problem evolve itself.

3.3 Cellular Automata and Lattice Gas Dynamics

Abbott and Minns (1997, pp. 460 *et seq*) discuss the applications of cellular automata (CA) to the problem of turbulence modelling. The study of hydrodynamics using cellular automata involves the study of fluid flow at a 'molecular' level. The fluid is supposed to consist of regular particles which move along lines defined by a regular grid or 'lattice' (*see* Boon and Noullez, 1987). At each time step, each of the particles usually moves along one link of the lattice. Each grid point contains one value out of a finite set of discrete values. The value in each gridpoint can be updated at each time step by following a simple, logical (but numerically expressible) rule depending upon the current value of the site itself and the value of neighbouring sites. Every site of the grid is effectively updated simultaneously. Collisions between particles are described by simple rules that are related to the basic rules of Newtonian (and indeed, strictly speaking, pre-Newtonian) physics and the conservation of mass. Cellular automata exhibit complex, random behaviour at individual sites but produce smooth, macroscopic-average behaviour which may closely resemble the descriptions of continuous systems as described by partial differential equations (Wolfram, 1986).

Through the choice of a suitable lattice and suitable rules, the averaging of the particle motions can provide a description of a fluid with properties identical to those of a fluid described by the Navier-Stokes equations (Wilson, 1988). In most current applications the rules are given by the human agent, but as so much current research in artificial intelligence is directed to generating or 'inducing' rules from observations using machine-learning techniques, it is to be expected that rule induction will proceed also in this case from on-line data streams (*see ,already,* Donald, 1994).

Frisch *et al.* (1986) demonstrated that only a lattice of equivalent triangles possesses the necessary symmetries in order to reproduce hydrodynamic behaviour correctly in two dimensions. This produces an hexagonal grid as shown in Fig. 3.2.

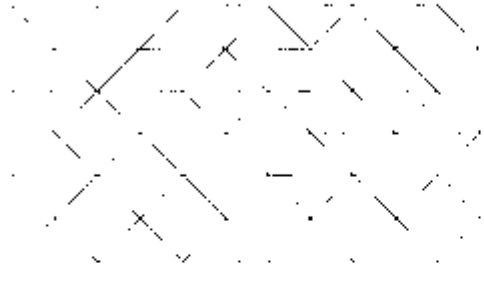


Fig. 3.2 Hexagonal lattice proposed by Frisch et al. (1986)

Possible collisions rules of the Frisch model are presented in Fig. 3.3.

Fig 3.3 Different possible collisions in a node

The properties of the fluid are calculated by averaging the motions of the particles over large areas or volumes of the simulation to obtain the average momenta of the flow for that part of the fluid, as shown in Fig. 3.4.

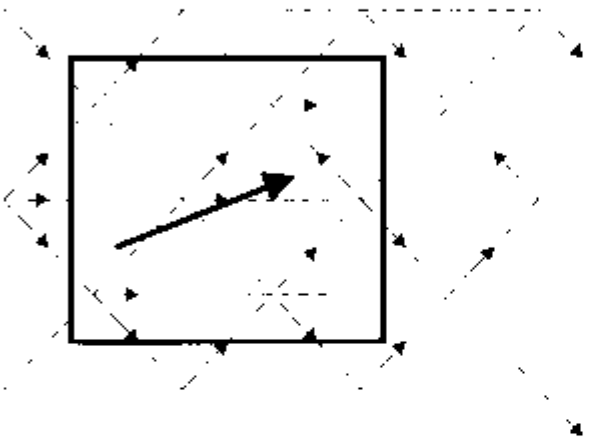


Fig. 3.4 Spatial averaging of moving-particles

Cellular automata models have some conventional advantages over standard computational-hydraulic codes: firstly, stability is not a problem for codes using lattice gas methods; secondly, boundary conditions are easy to implement; and, lastly, lattice gas operations are bit-orientated rather than floating-point-number-orientated and are therefore more suitable for computation on digital machines.

In another application area of cellular automata, the cells in the lattice are stationary and each cell can be in one of a finite number of states - a so-called *finite state automaton* (*see* Wuensche and Lesser, 1992). Transitions from one cell's state to another are determined through the application of some *local* rules. Thus, the current state of a cell depends solely upon the state of the cell at the previous time step and the states of the cells in the immediate neighbourhood (e.g. adjacent cells) also at the previous time step. As in any dynamical system, the system's variables change as a function of their current values. In a similar way to the lattice gas approach, the entire lattice of cells is effectively updated simultaneously.

The pattern of values across the whole array is the CA *global state* at a given time. The CA evolves through a succession of global states, constituting a *trajectory*, by the iteration of the global updating procedure, which provides a *transition function*. It can be shown that, provided

the CA from its initial global state is uniquely determined. When the transition function is restricted to a *local neighbourhood* (i.e. the neighbouring cells are all local to the target cell) we refer to a *local transition function* and the CA is said to have a *local architecture*.

Von Neumann (1949) first proposed the use of CA to model self-reproduction using a local, two-dimensional architecture consisting of 29 cell values. Since then, the trend has been to investigate the most simple possible architectures that are still capable of producing an overall complex behaviour. Perhaps the best known example is Conway's 'game of life', that is simply a local, two-dimensional CA with a 9 cell neighbourhood consisting of the target cell and the eight adjacent cells in a square grid. The values in each cell are simply binary.

The principal current advantage of CA and lattice gases is, however, their adaptability to machine learning. On the other hand, the immense effort now being put into the various branches of evolutionary algorithms would seem to imply that cellular-automatic methods will be overtaken on the machine learning side by these more complex and potentially 'richer' methodologies.

Goles and Martinez (1990) point out that cellular automata constitute a particular class of *automata network*. An automata network is defined simply as a graph, either finite or infinite, where each site or vertex takes states in a finite set and the state of a site changes according to a transition rule which only takes into account the state of the neighbours in the graph. They then explain that the McCulloch-Pitts neuron, depicted in Fig. 2.2, can also be considered to be a special class of automata network - that we generally refer to as neural networks. In this model, the graph is usually non-orientated and finite. The state set may be binary, {0,1}, or bipolar, {-1, +1}, and the transition rule is the threshold function defined by (2.2.1), which depends only on the neighbours and is weighted by real numbers, that is:

$$f\ (x_1,\ x_2,\ \ldots,\ x_n) \qquad H\left(\sum_{i}^{n} w_i x_i \ -\ \theta_i\right) \qquad (3.3.1) \text{--} (2.2.1)$$

where $H(\ \ldots\ )$ is the Heaviside step function, defined by (2.2.2), for binary states, or the sign function for bipolar states.

4 Rainfall-runoff Modelling

4.1 Introduction

Many of the experiments described in this chapter are taken from previously published papers of Hall and Minns (1993), Minns (1996) and Minns and Hall (1996, 1997).

Ever since the advent of what Linsley (1967) once referred to as the computer age in hydrology, considerable time and effort has been expended upon the problems of modelling the relationship between rainfall and streamflow. The variety of such models is legion, and the labels by which they are classified - lumped or distributed; conceptual or physically-based; single-event or continuous simulation; and so on - continue to proliferate. The directions in which modelling activity has been directed in recent years has been prompted largely by the rapid developments in powerful personal computers and workstations. The ease with which such machines can cope with large sets of ordinary and partial differential equations has stimulated interest in modelling both temporal and spatial variations in the physical processes, by which processes occurring in the atmosphere, and which subsequently provide rainfalls, are transformed into all manner of processes occurring in the land phase of the hydrological cycle, such as river flows. However, the full implementation of such models requires the use of large amounts of data that are necessary both to calibrate and to verify the model, and extensive parameter sets that must be manipulated for these purposes.

Perhaps the most widely-known of the modern generation of physically-based, distributed catchment modelling systems is the *Système Hydrologique Européen* (SHE), the original structure of which was described by Abbott *et al* (1986). Details of a case study in which SHE was applied to a river basin of some 4955 km^2 in India have recently been provided by Refsgaard *et al* (1992). Those authors provided a frank discussion of the substantial data requirements and supplementary fieldwork required to implement this model, and acknowledged that their use of 2 km × 2 km grid squares still did not provide a fully physically-based and fully-distributed description of the basin, even though it was entirely sufficient for the practical application in question. There remained a certain degree of empiricism in the representation of particular hydrological processes, even in these systems, so that process identification and the associated determination of parameter values by direct measurement continued to necessitate the use of extensive calibration procedures.

From this it can be concluded that for many problems of rainfall-runoff modelling involving, for example, record extension or forecasting, without any significant changes in land use or other such factors and over a certain range and distribution of antecedent soil conditions, simpler models would in most situations be equally accurate and much cheaper to apply. To be fair to the distributed, physically-based models, however, it should be pointed out that these are directed mostly to quite other processes than simple rainfall-runoff, and indeed from the point of view of

practical applications, problems of waste disposal, erosion, changes in vegetation and so on, are much more important than rainfall-runoff alone. Even so, and as will be explained subsequently, ANNs in themselves may still be useful in such connections also.

In contrast, systems investigations, which Amorocho and Hart (1964) regarded as being concerned with the direct solution of technological problems subject only to the constraints imposed by the available data, and so not subject to 'physical' considerations, has recently undergone something of a renaissance, largely through the adaptation of artificial intelligence techniques, such as artificial neural networks (ANNs) and genetic algorithms. The particular advantage of the ANN is that, even if the 'exact' relationship between sets of input and output data is unknown - but is still acknowledged to exist - the network can be trained to learn that relationship, requiring no *a priori* knowledge of the catchment characteristics.

4.2 A Simple Laboratory Catchment

For the initial phase of experiments, data were required from a simple hydrological system, preferably having volume continuity between rainfall and runoff, i.e. minimum losses between the rainfall and the runoff, so that the problem concerning antecedent conditions does not arise. Hall and Minns (1993) describe the results of their experiments using data from a simple laboratory catchment. The laboratory catchment had an impervious surface, and sample data were obtained from a number of different studies in the tabulations of urban catchment data provided by Maksimovic and Radojkovic (1986). The latter include some of the experiments from a joint project between the US Soil Conservation Service and the US Public Highways Administration, subsequently reported by Izzard and Augustine (1943) and Izzard (1946).

Four events relating to a surface of clashed slate roofing paper, 1.83 m wide by 14.63 m in length and sloping at 0.005, were selected for study. The rainfalls consisted of pulses of constant intensity, including changes from 'high' to 'low' intensity (event 127) and 'low' to 'high' (event 128), and double bursts of 'high' intensity rainfall with an intervening period of no rain, with the bursts of 'long' (event 129) and 'short' (event 130) duration. Since the recession data for event 129 was incomplete, this storm was reserved for verification, and events 127, 128 and 130 were used for training. As presented by Maksimovic and Radojkovic (1986), the data were recorded at a variety of time intervals from 5 seconds upwards, and so linear interpolation was employed to produce rainfall and flow ordinates at regular time intervals. The original study of Hall and Minns (1993) used data with time intervals of both 5 seconds and 10 seconds for different experiments. The use of 5 second data did not appear to provide significantly better results than the 10 second data so it has been decided to reproduce these experiments here using only the 10 second data. This editing produced 912 sets of data for training and 108 sets of data for verification.

Although only representative of the most simple hydrological systems, such laboratory catchments display certain hydrodynamic features, of which the most obvious is the anomalous, so called, *pip*, i.e. the sudden rise and fall of flow above the equilibrium discharge that occurs when rainfall ceases. As explained by Yu and McNown (1964), these pips result from the extra surface roughness generated by the beating of raindrops into the thin sheet flow across the surface

of the laboratory catchment. When the rainfall terminates, the depth of flow momentarily exceeds that which can be supported by the rugosity of the surface without rain, and the extra depth is eliminated quickly in the form of a sudden rise and fall in flow, which appears on the discharge hydrograph as the 'pip'. Although of little significance hydrologically, these pips form an obvious feature of the pattern of catchment response on which the neural network is to be trained.

When applying a neural network to the rainfall/runoff problem, the stimulus is obviously the rainfall and the response is the streamflow at the basin outlet. Since the flow at any instant is effectively composed of contributions from different sub-areas whose time of travel to the outlet covers a range of values, both the concurrent and antecedent rainfalls can be considered as stimuli. The initial network configuration therefore consisted of M input nodes with the rainfall ordinates for time t and the (M - 1) previous time intervals, and one output node with the flow at time t. In all cases, only one hidden layer of nodes was incorporated into the network, the number of nodes in the hidden layer being chosen arbitrarily to give about 60 per cent of the number of nodes in the input layer. For convenience of physical interpretation, the number M will be termed the *window length*.

Training of the neural network was continued until the global error of the network, as based upon the sums of squares of the differences between observed and computed values, was brought down to an acceptable level. In the majority of runs this meant that the training was continued until the number of data sets presented to the network had exceeded 10^6. Since the global error as implemented in the software package employed was dependent upon the number of nodes in the network, a more general fitting criterion was sought. As the review by Diskin and Simon (1977) has shown, a variety of such indices have been applied in hydrological modelling, but perhaps the form that has been used most widely is the *coefficient of efficiency* defined as one minus the quotient of the mean square error and the variance of the observed flows, i.e.

$$F = 1 - \frac{\frac{1}{m}\sum_{i=1}^{m}(q_i - \hat{q}_i)^2}{\frac{1}{m-1}\sum_{i=1}^{m}(q_i - \bar{q}_i)^2} \qquad (4.2.1)$$

where $\hat{q}_i$ are the model estimates of the flow ordinates, q_i, $i = 1,2,\ldots,m$, and $\bar{q}_i$ is the mean of the q_i. As will be explained later however, (*see also* Hall, 1997) there are still some objections to using this measure in the present case.

The first question to be answered when applying the ANN was the effect of changing the window length. Runs were therefore performed with windows ranging from 5 to 50 time intervals in length. The results obtained are typified by those presented in Fig. 4.1 (a), (b) and (c) which, for the verification event 129, compare the reproduction with 10-, 25- and 50-interval windows respectively.

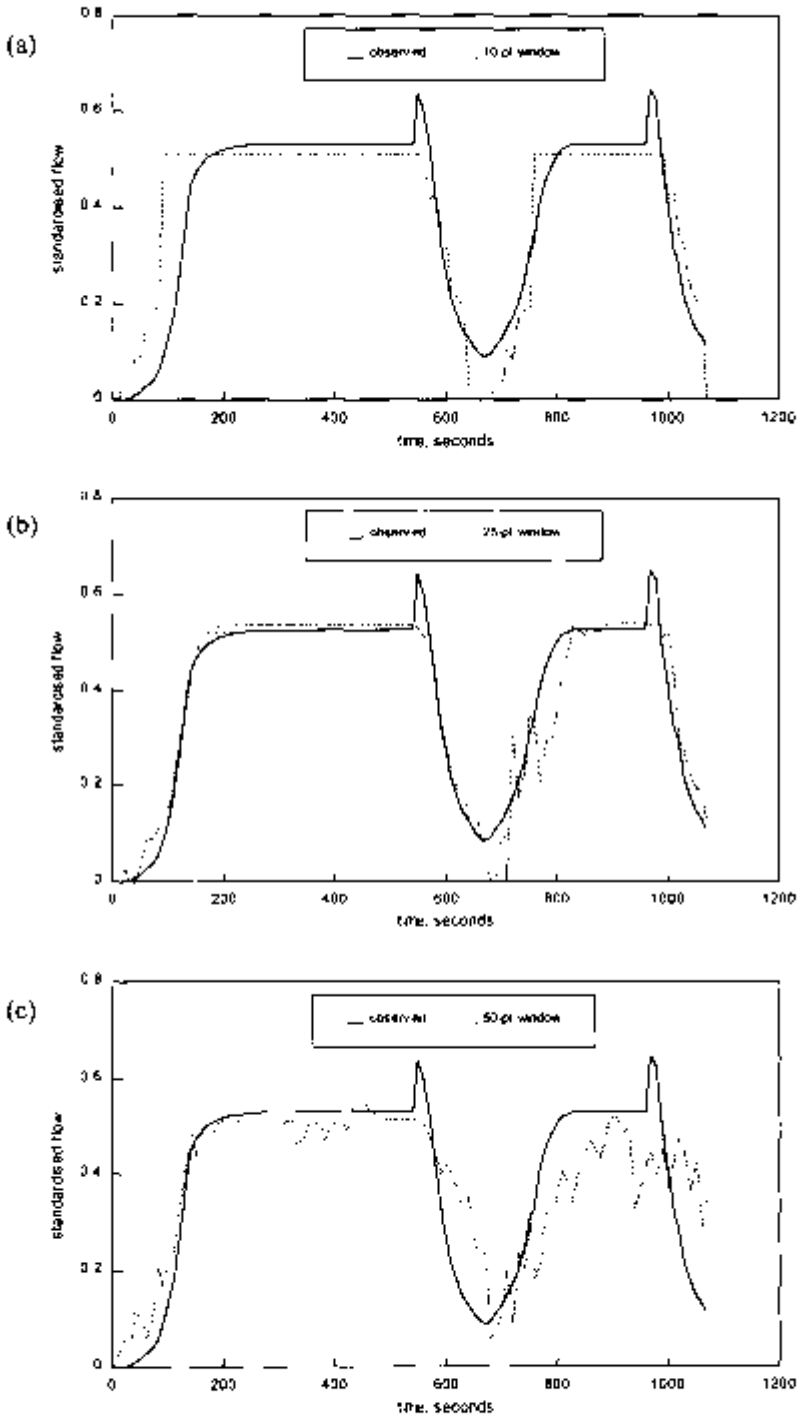


Fig. 4.1 Performance of neural networks on the verification event for a laboratory catchment using only 10-second-interval rainfall data as input for (a) 10-interval; (b) 25-interval; and (c) 50-interval windows

With the 10-interval-or-less windows, the hydrograph shape is reproduced quite poorly. The square shape of the output hydrographs seems to represent only the shape of the rainfall bursts and there is a very poor representation of the rising limbs and the recession limbs. The equilibrium discharges are also underestimated. The 25-interval window provides a hydrograph which follows the first burst quite closely but is less successful with the second. Indeed, the second half of the event displays a significant disturbance on the rising limb, which has been introduced some 25 time intervals after the cessation of the first rainfall burst.

When the window length is increased to 50 intervals several developments can be observed. Firstly, the number of connections in the ANN increases exponentially, which increases the training time considerably. The training in this case is further frustrated by the presence of local minima. Secondly, if the number of hidden nodes in the network is decreased significantly to improve the training performance (e.g. 15 hidden nodes were used for the results in Fig. 4.1), then the verification results demonstrate a serious degradation in the performance of the ANN. The ANN can no longer 'generalise' the relationship between rainfall and runoff. There is simply *too much* data being presented at the input layer and - to maintain the biological analogy - the ANN gets 'confused'. The coefficients of efficiency for the training and verification of the various ANNs mentioned above are given in Table 4.1.

Table 4.1 Coefficients of efficiency for ANN models shown in Fig. 4.1

No. of rainfall inputs	training sequence (events 127, 128 & 130)	verification sequence (event 129)
10 pt window	0.911	0.783
25 pt window	0.989	0.926
50 pt window	0.966	0.729

The window length can obviously be too long as well as too short. The 25-point window appears to give the best results in both training and verification. It is essential to stress here the importance of the verification data sequence in selecting the most appropriate ANN configuration. The performance of the ANN on its own training data sequence, which are expressed as coefficients of efficiency in Table 4.1, do not reflect the inherent shortcomings of the trained ANN as evidently as do the verification results. An ANN will quite often demonstrate an outstanding ability to learn relationships from any training data sequence to a very high degree of accuracy (e.g. coefficient of efficiency above 0.9), even if these relationships are in fact non-deterministic or even nonsensical. The learned relationships must first be validated through the application of the verification data sequence to the trained network, before anything at all can be said about the generalisation properties of the trained ANN to new or 'unseen' data.

In the above example, 25 points of 10-second data represent a time interval that broadly encompasses the range of centroid-to-centroid lag times of the training events, a result that has some intuitive appeal. However, the network having this input fails to reproduce the observed anomalous pips, and introduces some significant noise in the second half of the verification event. Since increasing the window length will not eliminate the latter feature, some additional information must be provided to the network in order to allow the observed and spurious features to be distinguished.

The problem of the spurious noise on the second rising limb of the verification event is, in fact, caused by contradictory information being presented to the network under 'no-rainfall' conditions. Before the storm begins, all of the antecedent rainfall inputs within the window are zero and the flow output is also zero. One window length after the end of the rainfall storm, the rainfall input again consists of zeros but the flows are in recession and outputs are most definitely non-zero. The network has no information to discriminate between these two 'no-rainfall' conditions and once more becomes 'confused'. This extra information could be provided, for example, by a binary variable (e.g. unity for post storm; zero for pre-storm conditions). However, the antecedent flows themselves provide an indication as to whether rain has occurred or not. In addition, such flows add the further information that the longer the interval of zero input, the more the output decreases. This approach of including an output variable in the input constitutes a form of *recurrent back-propagation* (*see, for example*, Hertz *et al*, 1991, pp. 163 *et seq*).

The inclusion of the flow at time $(t - 1)$ as an input to determine the flow at time t may appear to introduce an element of flood routing into the model, but that is not the purpose of the ANN. Unlike the conventional rainfall-runoff model, the network seeks to learn patterns and not to replicate in detail the physical processes involved in transforming input into output. The learning process does not depend upon any assumptions relating to the form of the input-output transfer function, the number of (active) parameters or their possible physical interpretation. In the terms of the discussion by Amorocho and Hart (1964), the ANN could perhaps be regarded as the *ultimate black-box model*.

This introduction of recurrent back propagation is justified as shown in Fig. 4.2, which depicts the results of a network that receives 25 rainfalls and 4 antecedent flows as input.

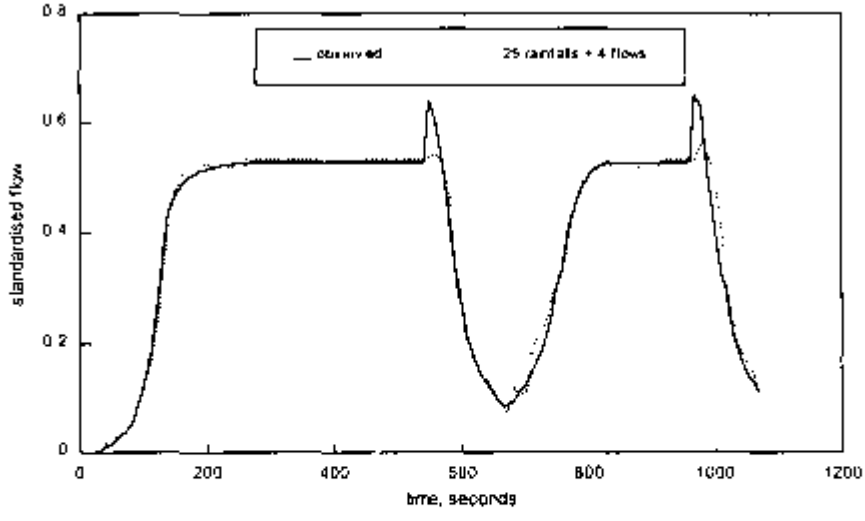


Fig. 4.2 Performance of neural network on verification event for laboratory catchment using 25 rainfall ordinates and 4 antecedent flow ordinates as input

Fig. 4.2 indicates that the spurious noise has been almost been entirely eliminated from the second half of the verification event and that the equilibrium discharges of both rainfall pulses are successfully captured by adding the antecedent flows. The coefficients of efficiency for the training and verification data sets are now 0.998 and 0.985 respectively. Perhaps even more satisfying is the appearance of the two anomalous pips in the network output, although their timing and magnitude are somewhat lacking in agreement with the observed pips.

The results of the above experiments clearly show the effect of the choice of input data upon the ability of the ANN to predict discharges. These tests have considered until now either just rainfall or a combination of rainfall and discharge in the input array. As a logical extension to these tests, one final experiment was carried out that did not make use of any rainfall data, but used only antecedent flow values as input to the ANN.

Fig. 4.3 depicts the results of a network that uses only the 5 antecedent flow values to calculate the current flow value. The coefficients of efficiency for the training and verification of this ANN model are 0.993 and 0.967 respectively.

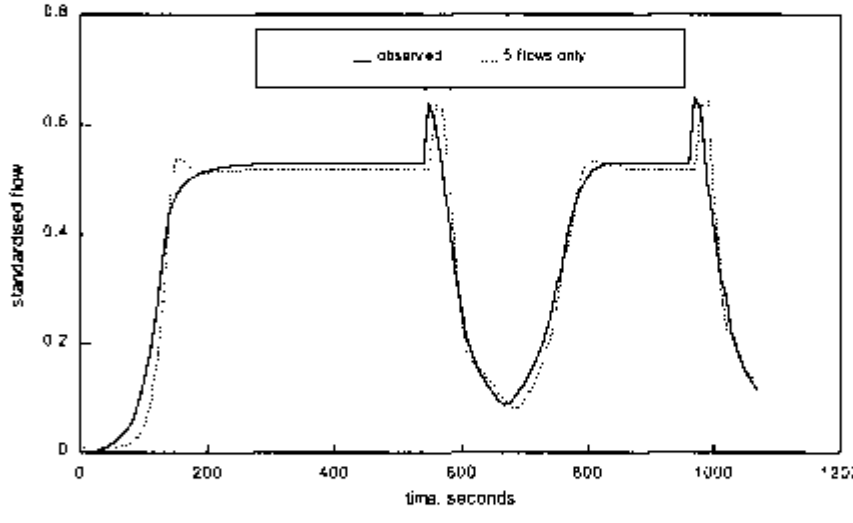


Fig. 4.3 Performance of neural network on verification event for laboratory catchment using 5 antecedent flow ordinates only as input

Although the rising limb of the verification event is not reproduced very accurately and the equilibrium flows are slightly underestimated, this ANN model provides only slightly poorer performance than the 25 rainfall and 4 flow model used earlier. The overshoot at the top of each rising limb is caused by the fact that the network has no other information available that tells it at which level the rising limb should stop until the actual measurements indicate that this is so. That is, at the top of the rising limb, the output from the ANN wishes to continue rising in magnitude based only upon the pattern of the preceding flows. It is not until several time steps have passed for which the measured values are all constant, that the ANN 'recognises' that the equilibrium level has been reached. Similarly, the phase error that occurs in the timing of the

anomalous 'pips' is caused by the fact that the ANN has no knowledge about the cessation of the rainfall until one or two time steps after the actual measured values start to decrease.

The results of this experiment indicate that the ANN is also a very powerful device that could also be used for time-series extrapolation. Perhaps one of the most well known devices for sequential data assimilation and time-series extrapolation is the Kalman Filter (Kalman, 1960). This approach is a statistically optimal method for the sequential assimilation of data into linear models. The Kalman filter provides an estimate of the state of the system at the current time step based on all measurements of the system available up to and including the current time in very much the same way as the above ANN model (*see*, for an application in hydrodynamic modelling, Cañizares *et al*, 1996).

4.3 A Small Sewered Catchment

The results with the laboratory catchment data were sufficiently encouraging to initiate a subsequent phase of testing with field as opposed to laboratory data. In order to minimise the possible effects of seasonal variations in losses, these further tests were carried out on records from an urban catchment area. The Cantley Estate in Doncaster was gauged for a 3-year period in the late 1950s as part of a research programme carried out by the then Road Research Laboratory (see Watkins, 1962). The catchment, which has a gross area of 5.14 ha, is served by a separate surface water drainage system having an outfall 610 mm in diameter. The details of 16 storm events were kindly supplied by the Institute of Hydrology, Wallingford. 12 of these events were randomly selected for training and the remaining 4 reserved for verification. With data at one-minute intervals, there were therefore 985 data sets for training and 270 for verification. The results of the best-performing network configuration are presented in Fig. 4.4.

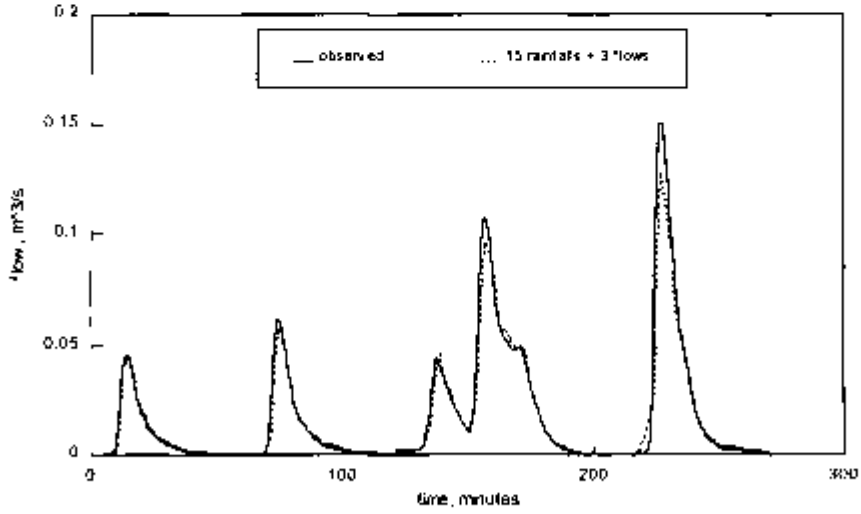


Fig. 4.4 Performance of ANN on verification events for Cantley Estate sewerage system using 15 rainfall ordinates and 3 antecedent flow ordinates as input

Several different network configurations were trained and tested before arriving at the results given in Fig. 4.4. As with the laboratory catchment data, a series of runs was carried out exploring the effect of changing the length of the rainfall window, and then adding antecedent flows to the input. Similar results were obtained as with the laboratory catchment. Rainfall-only input produced a very noisy output as before, with peak flows significantly underestimated on some events and overestimated on others. In addition, the lower limbs of the recessions were too steep. The addition of 3 antecedent flows removed most of these undesirable features, although the highest peak flow rate was both underestimated in magnitude and late in timing.

Having demonstrated that a neural network with a suitable choice of inputs is capable of reproducing, with some fidelity, the responses to storm events upon which it has not been trained, the question arises as to whether the approach offers any advantages over a conventional black-box rainfall-runoff model. The 4 verification events from the Cantley Estate have therefore been modelled separately by means of a well-established, conceptual hydrological modelling package, RORB (Mein *et al*, 1974).

The basic element of the RORB model is a single, conceptual, non-linear reservoir for which the relationship between storage, S, and discharge, Q, is given by:

$$S = K_c K_r Q^m \tag{4.3.1}$$

where K_r is a storage constant applicable to all sub areas within the catchment and K_r is a relative delay time applicable to individual channel reaches within the network estimated from the expression:

$$K_r = f \frac{L_i}{L_{av}} \tag{4.3.2}$$

where L_i is the length of the reach represented by the storage element, L_{av} is the average flow distance of sub-catchment inflows within the channel network, and f is a factor depending upon the type of channel reach, i.e. natural, lined or unlined.

For this experiment, the power of the non-linear reservoir, m, was set to the default value of 0.8, and the initial loss and storage constant manipulated until the peak flow rate and total runoff volume were satisfactorily reproduced. In the case of the event of 3 July, 1957, which was double-peaked, the rainfall was separated into two bursts, thereby introducing the ratio between the runoff volumes caused by each burst as a third calibration parameter. Since the RORB model was calibrated for each event individually but the neural network operated on all 4 events with the *same set* of weights determined from the training, this comparison is inherently unfavourable to the ANN model.

The plots of the single-peaked storm of 26 August, 1956 and the double-peaked event of 3 July, 1957 are displayed in Figs 4.5 and 4.6 respectively.

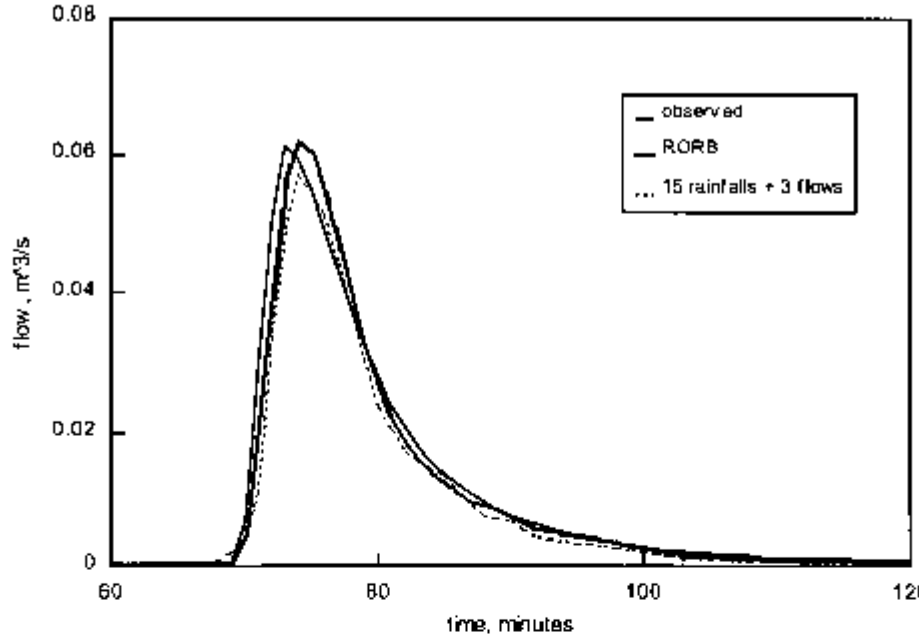


Fig. 4.5 Model comparison for storm of 26 August, 1956 (Cantley Estate)

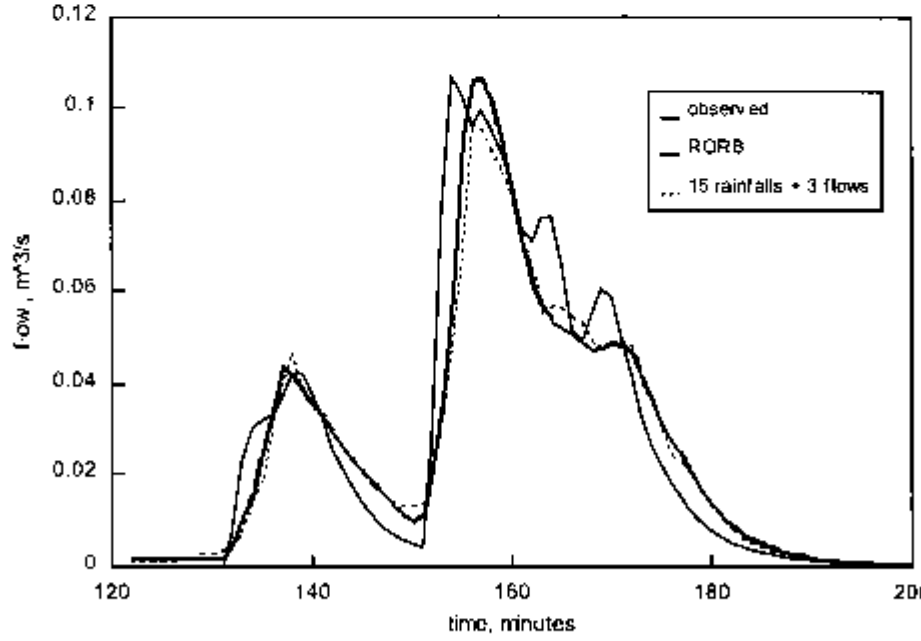


Fig. 4.6 Model comparison for storm of 3 July, 1957 (Cantley Estate)

The results are compared in Table 4.2 in terms of their coefficients of efficiency. The coefficients of efficiency of the two models on these four events are generally comparable in magnitude. In the case of the event of 3 July 1957 (that with a pronounced double peak), the performance of the neural network model is obviously superior.

Table 4.2 Comparison between fit provided to four storm events by an ANN model and the RORB conceptual model in terms of coefficients of efficiency

Storm of	Coefficients of efficiency	
	RORB	ANN
23 August 1956	0.974	0.981
26 August 1956	0.983	0.982
3 July 1957	0.884	0.978
21 July 1957	0.990	0.951

Some important factors should be considered when evaluating these results. Firstly, the calibration parameters for the RORB model included an initial loss rate, while application of the neural network did *not* involve any consideration of loss separation. Moreover, the neural network has no calibration parameters as such, but only the set of weights which it learns itself. It thus involves no operator intervention and no *a priori* knowledge of the catchment. Although the training of the neural network requires a substantial investment in computer time, the procedure is far more straightforward than is the calibration of even a simple conceptual model, which must be undertaken on an event-by-event basis.

The results obtained are sufficient to demonstrate that, for situations involving rainfall-runoff modelling in which there are no extraneous influences such as land-use changes, a neural network has the potential to perform in a comparable fashion, if not better than, a conceptual hydrological model.

4.4 Linear and Non-linear Catchments

The next important consideration is the applicability of ANNs to more complex 'real world' catchments. Although the standard solution algorithm for ANNs will achieve convergence for almost any problem, it would appear that the most simple ANN architectures have more difficulty in learning more non-linear relationships. This section therefore describes a series of numerical experiments that were undertaken by Minns and Hall (1996) with the specific purpose of evaluating the performance of ANNs on rainfall and runoff data from theoretical catchments exhibiting a range of behaviour patterns varying from the linear to the highly (in hydrological terms) non-linear. Owing to the virtual impossibility of collecting hydrometric data from catchments that could be classified *a priori* as either linear or non-linear, but were otherwise identical in catchment characteristics and input rainfall patterns, the conceptual hydrological modelling package RORB (Eq. 4.3.1) was employed to generate streamflow responses from a synthetic time series of storm events for representative (linear and non-linear) catchments.

In this manner, the ANN could be tested solely on its performance in learning the (linear or non-linear) relationship between rainfall and runoff, all other factors being regarded as equal. The precise form of the model used to generate the runoffs from the rainfalls is of little importance as the ANN is not being applied to identify this model but principally to produce responses typical of those encountered in hydrological practice.

4.4.1 Generation of rainfall data

For the purposes of the numerical experiments, sequences of storm events of varying duration, total depth and profile, occurring at irregular intervals, were required that could be routed through simple conceptual hydrological models with different degrees of non-linearity in order to produce the corresponding streamflow outputs. For simplicity, these rainfalls were treated as areal averages. Since several storm sequences were required, they were produced using Monte Carlo methods based on the following assumptions:

1. storm durations were normally-distributed, with a mean of 20 h and a standard deviation of 6 h;
2. storm rainfall depths were lognormally-distributed, with a mean of 25 mm and a standard deviation of 2 mm. (These statistics imply that the distribution of depths had a coefficient of variation of 0.785 and a skewness coefficient of 2.84);
3. the shapes of the six storm profiles could be described by simple polynomial functions, broadly based on those of the UK Flood Studies Report (Natural Environment Research Council, 1975), and including early-peaked and late-peaked as well as symmetrical events (with a constant intensity profile also included as an extreme case); and
4. the inter-event times were taken as double the previous storm duration minus one hour.

Initially, three sequences of 14 storm events with hourly data were generated, the profile shapes being selected by sampling from a distribution uniform over the range one to six. The first was a training sequence with a total duration of 764 h. Five of the six profiles were represented, with durations having an average of 19.2 h and a standard deviation of 6.95 h. The average depth was 31.6 mm, with a standard deviation of 1.9 mm. This sequence is referred to as RAIN1 in Table 4.3. The other two sequences were employed for verification purposes. The first of these verification data sets was generated in such a way that the maximum values all fell within the range defined by the training sequence. This sequence was used for verification of the ANN under 'normal' rainfall conditions. However, if an ANN were to be applied to a real catchment, even if the training data included all the available measurements, there remains always a small but non-negligible probability that an extreme event beyond the range of recorded experience might occur in the future. A second verification sequence was therefore generated that contained rainfall maxima outside the range of those upon which the training data had been based. This sequence was used for verification of the ANN under 'extreme' rainfall conditions.

Table 4.3 Some properties of the generated rainfall data sequences

Rainfall data sequence	Used for:	Total duration of sequence (h)	Average depth of storms (mm)	Standard deviation (mm)	Maximum intensity (mm/h)
RAIN1	training	764	31.6	1.9	5.2
RAIN2	normal verification	794	24.6	2.1	6.6
RAIN3	extreme verification	794	24.3	3.3	13.0

The two verification sequences each had a total duration of 794 h and all profiles were represented. In the first data set, individual events had an average duration of 19.8 h and a standard deviation of 4.9 h, and a mean depth of 24.6 mm with a standard deviation of 2.1 mm. This sequence is referred to as RAIN2 in Table 4.3. The second verification sequence was constructed by employing the same seed as that employed for the first, but assuming that the storm depths were lognormally-distributed with a mean of 25 mm and a standard deviation of 3 mm, which implied a coefficient of variation of 1.53 and a skewness coefficient of 8.2. The actual mean and standard deviation of storm depths produced was 24.3 mm and 3.3 mm respectively. This sequence is referred to as RAIN3 in Table 4.3.

4.4.2 Generation of the runoff hydrographs

According to Laurenson and Mein (1988), the exponent used in the RORB model (4.3.1) is rarely less than 0.6 or greater than 1.0 when modelling catchment runoff response to rainfall, and a trial value of 0.8 is recommended on beginning a modelling exercise. A brief review of the available literature shows that the values adopted for exponents has ranged from 0.67 (Watt and Kidd, 1975) to 0.8 (Selvalingham *et al*, 1987), with a predominance of values between 0.7 and 0.8 (Laurenson, 1964; Askew, 1970; Mein *et al*, 1974; Hong and Mohd Nor, 1988). Three models of theoretical catchments were therefore adopted to cover the entire range of possible physical catchment behaviour:

(i) $m = 0.8$ to represent the typical non-linear relationships encountered in practice, and referred to here as a *regular* catchment;
(ii) $m = 1.0$ to represent an extreme *linear* catchment; and
(iii) $m = 0.5$ to represent an *extremely non-linear* catchment..

In order to run the RORB software, a hypothetical catchment area and main channel length had to be assumed in order to establish the value of K_c. The chosen values of these characteristics were consistent with those of a rural drainage area of about 30 km^2 in southern England. Although these considerations are not particularly relevant to the learning of patterns, the size of the catchment broadly determines how many antecedent rainfall depths are required in developing the ANN and therefore influences the overall size of the network. For simplicity, no losses were separated and the catchment was considered to have no impervious area. The K_c value was set to 20. The time series of flows so obtained reflected very well the range in response characteristics represented by the three theoretical catchment models, with the extremely non-

linear catchment model showing rapid rises and recessions in contrast to the slow rises and sustained recessions of the linear catchment model. For the purposes of illustration, the rainfall hyetographs and flow hydrographs generated by the regular catchment model are presented in Fig. 4.7.

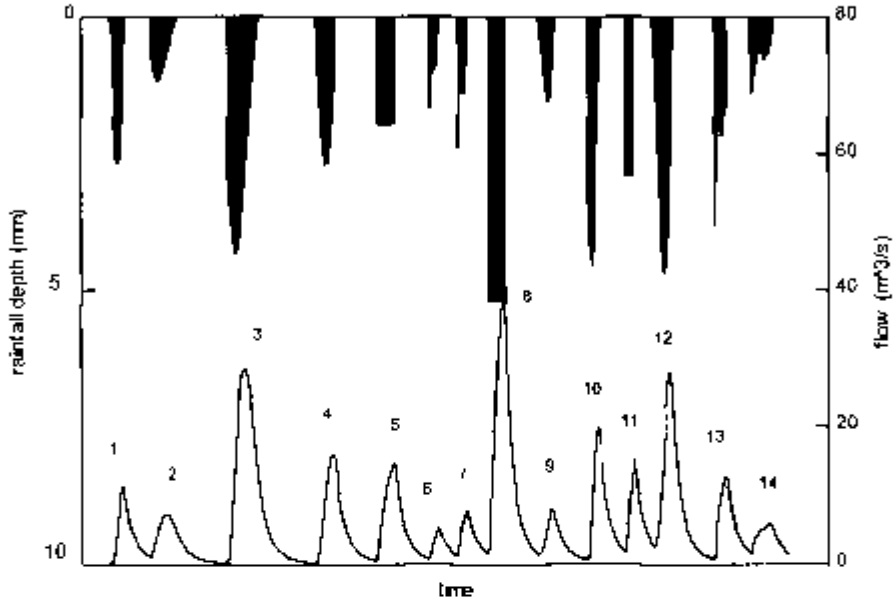


Fig 4.7 Rainfall and runoff data for the regular theoretical catchment model

For each catchment model, three data sets were generated corresponding to the three rainfall data sequences in Table 4.3. The maximum flow rates generated by each catchment model are summarised in Table 4.4.

Table 4 4 Maximum flow rates generated by each catchment model for the rainfall data sequences of Table 4 3

Data set	Rainfall data sequence	Maximum flow rates (m^3/s)		
		Regular catchment	Linear catchment	Non-linear catchment
Training	RAIN1	39.35	27.10	45.14
Normal verification	RAIN2	37.27	27.04	54.40
Extreme verification	RAIN3	87.95	59.47	110.60

The rainfall-runoff data sets generated by RAIN1 are referred to as the *training* data sets, while RAIN2 was used to produce the *normal verification* data sets, and RAIN3 was used to produce the *extreme verification* data sets.

4.4.3 Standardisation of the data

As explained earlier (in §2.4), prior to presenting the data to the ANN for training, a standardisation must be applied in order to restrict the output data range to the interval of zero-to-one, corresponding to the limits of the sigmoid function (2.4.2) in the output nodes of the network. The significance of this standardisation should not be underestimated. If different standardisation factors were to be applied to the training and verification sequences, then the actual numbers represented by unity in the output node of the ANN would be different. That is, the user of the ANN would be assigning a different 'meaning' to the output than the one that was adopted by the ANN during the training process.

In practice, a trained ANN can only be used in the recall mode with data that is, in some way, of the same or similar type to the data that it has 'seen' before. An ANN generally performs very poorly when used for extrapolation. For example, if the maximum flow that the ANN has learned to predict is 50 m^3/s (corresponding to, say, an output of 1.0 from the sigmoid function) it is impossible for the ANN ever to predict a flow value exceeding 50 m^3/s (i.e. the sigmoid function cannot ever exceed 1.0).

The choice of the range for standardisation may therefore influence significantly the performance of the ANN. Standardisation or rescaling of the input data is not absolutely necessary for the standard MLP networks used here. The actual values of the input data only affect the magnitude and distribution of the weights connecting the input layer to the hidden layer. Input data with very large values will generally have very small weights associated with them and vice versa. This can significantly affect the speed and efficiency of the learning algorithm.

Furthermore, different ranges of values for different input variables may affect the sensitivity of the trained ANN to variations in one input value compared to another. For example, if an input data stream contains a large number of values that are very small in magnitude as well as several values that are very large, then a linear standardisation of this data will divide all of the values by a very large number corresponding to the maximum value. This may then result in standardised values of the smaller data points that are all very close to zero. Subsequently, the network will not be sensitive to this input or output. In this case a standardisation of the data using a logarithmic scale may be necessary.

In fact, the documentation of the WinNN software used in these experiments points out that the weighted-sum input to a sigmoid function should be between −2 and +2, otherwise the input will 'saturate' the neuron and cause it to output 0 or 1 all the time. In these experiments, the input rainfall and flow data were therefore rescaled between 0 and 3.

The output data, on the other hand, must always be standardised between 0 and 1, corresponding to the output range of the sigmoid. The sigmoid function is asymptotic to 0 and to 1 so that, in fact, it can never reach these values exactly. For this reason it is often desirable to standardise the output values of the network to the range 0.1 to 0.9, or even 0.2 to 0.8 (Tang & Fishwick, 1993; Smith & Eli, 1995).

For the experiments described here, the output data for both the training and verification data sets were standardised between 0.1 and 0.9, corresponding to zero and the maximum flow rate in the

training data set respectively. This meant that the maximum flow rates in the normal verification data sets (see Table 4.4) all fell within the same standardisation range as their associated training data sets except for the extremely non-linear catchment. In this latter case, the maximum flow rate in the verification data set was, in fact, slightly larger than the maximum flow rate in the training data set, resulting in a maximum standardised value in the verification data set of 1.085, which, of course, exceeds the upper limit of the sigmoid function of 1.0. Since this error only involved one or two of the 794 data points in the verification data set, it was not considered to be a significant problem here.

4.4.4 Training and verification experiments under 'normal' conditions

For all three conceptual models, experiments were carried out with a 3-layer ANN (i.e. one hidden layer). Based on the results obtained by Minns and Hall (1996) as well as some extra experiments, a network configuration was finally chosen that involved the use of the concurrent and 14 antecedent rainfall depths and three antecedent flow ordinates. 12 hidden nodes were eventually chosen for the hidden layer. The output consisted of the single, concurrent flow value.

In order to provide a visual impression of the degree of fit obtained during training and verification, two consecutive events from the training and normal verification sequences, which include the largest of the 14 generated storms in each case, have been selected for illustration. Figs 4.8 and 4.9 show the performance of the 3 layer ANN for the (a) moderately non-linear, or regular; (b) linear; and (c) extremely non-linear catchment models for the training and normal verification data sequences respectively. In all of the training cases, the hydrograph from the smaller event is well simulated, but the 3-layer ANN marginally underestimates the six or seven peak ordinates from the larger event. In the verification sequences, there is a similar underestimation of the peak ordinates in each case. In addition, a slight deterioration in the performance of the 3-layer ANN can be observed for the extremely linear and extremely non-linear catchments. This slight deterioration is confirmed in Table 4.5, which summarises the results from both training and verifying the 3-layer ANN on the data from each of the three models.

Table 4.5 Coefficients of efficiency for 3- and 4-layer ANNs fitted to rainfall and runoff series from three different conceptual hydrological models

Catchment model	Training sequence		Normal verification sequence	
	3-layer ANN	4-layer ANN	3-layer ANN	4-layer ANN
regular	0.9987	0.9983	0.9941	0.9953
linear	0.9985	0.9985	0.9904	0.9904
extremely non-linear	0.9990	0.9990	0.9891	0.9810

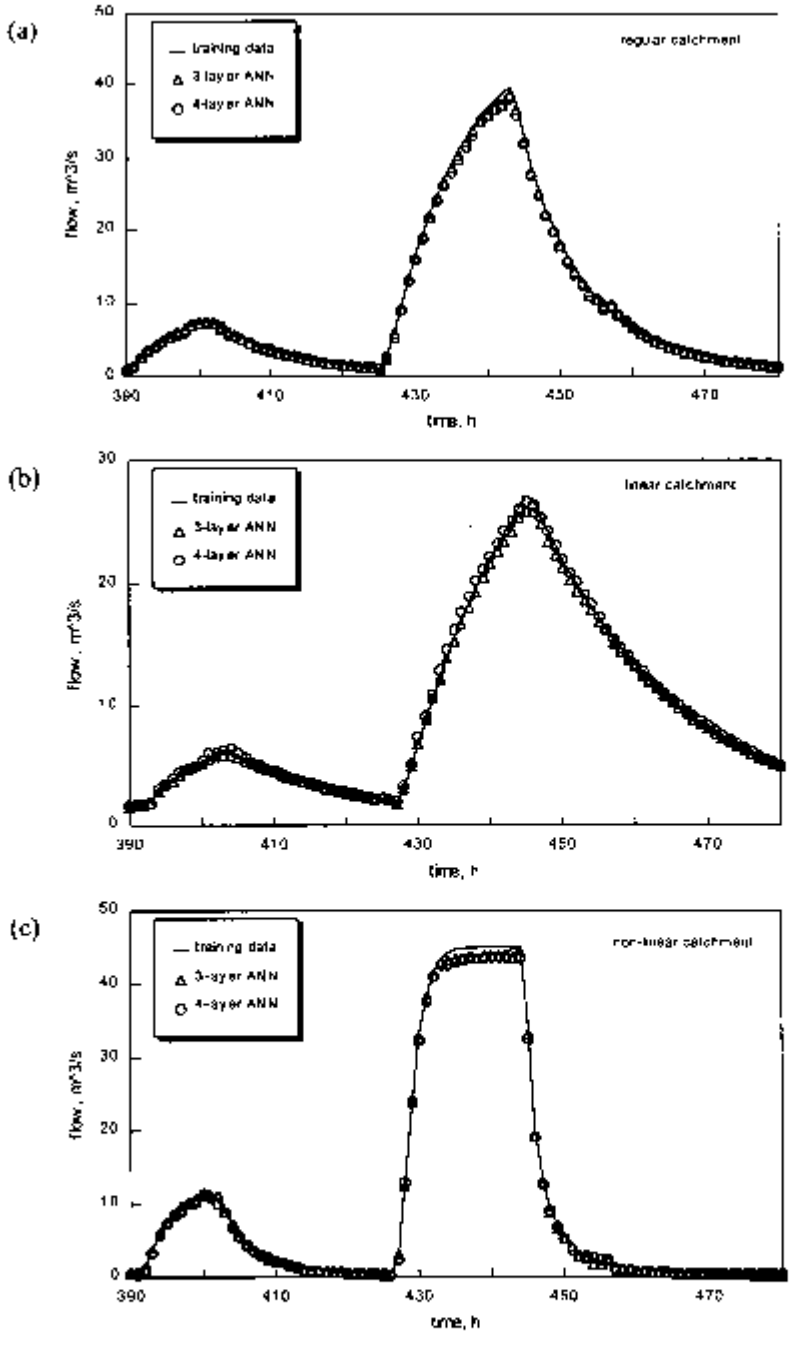


Fig. 4.8 Training of 3- and 4-layer ANNs on input and output from each of the three conceptual catchment models. (a) regular catchment, (b) linear catchment, and (c) extremely non-linear catchment. For clarity of illustration, two events only have been selected

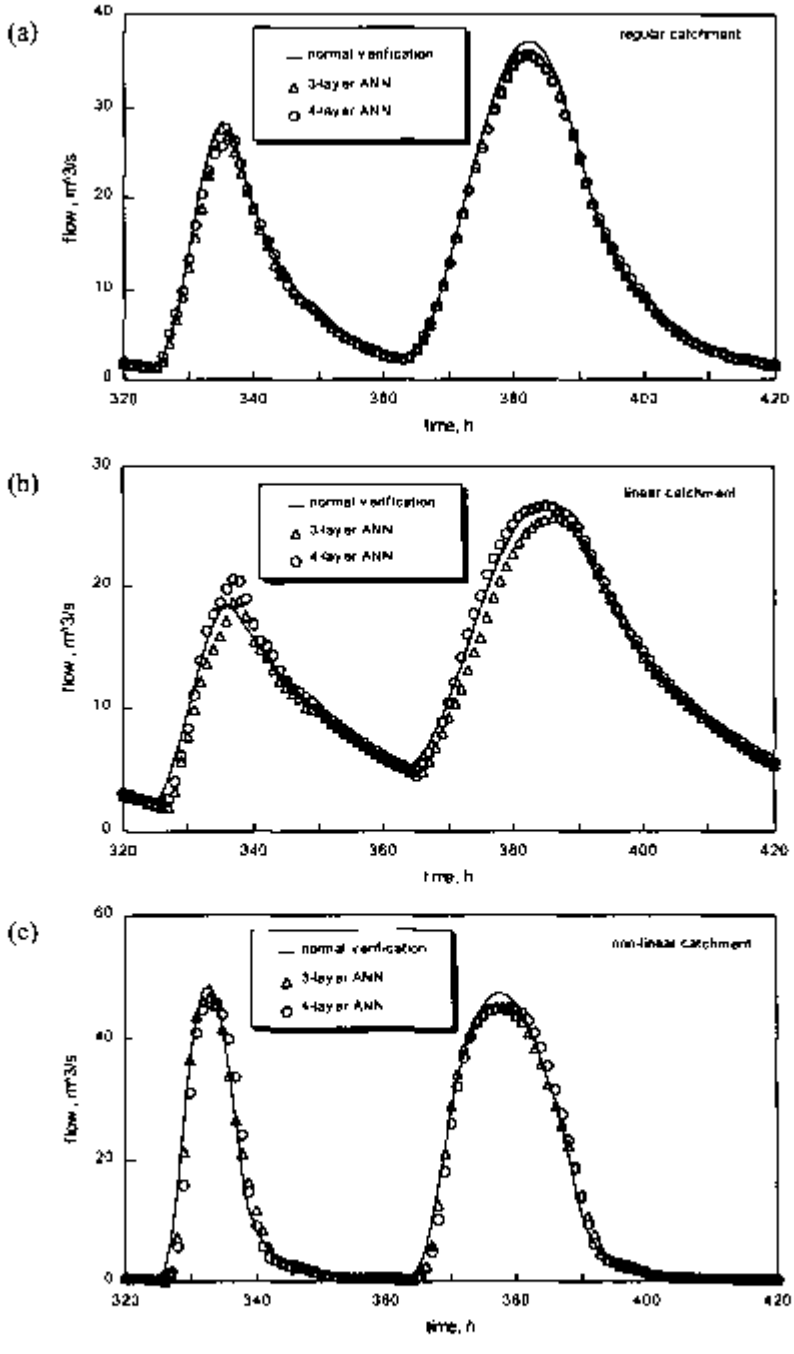


Fig. 4.9 Normal verification of 3- and 4-layer ANNs on input and output from each of the three conceptual catchment models: (a) regular catchment; (b) linear catchment; and (c) extremely non-linear catchment. For clarity of illustration, two events only have been selected

Table 4.5 shows that the goodness-of-fit obtained was such that the majority of the coefficients of efficiency varied only in the third or fourth place of decimals. In verification, the performance of the ANN on the regular catchment was marginally the best, although there was little to choose between that and the two other extreme cases. Since the network inputs included the flows at previous time steps, the ANN could be considered to be modelling the change in flows rather than their absolute values. In these circumstances, for the calculation of the coefficients of efficiency, the variance of the differences in flows, $q_t - q_{t-1}$, could be preferred to the variance of the observed flows in Eq. (4.2.1). However, investigation showed that, for the data sets employed in this study, the variance of the differences was usually of the order of 10^{-2} times the variance of the observed flows, but that the mean square error could be as high as 10 times the variance of the differences. In these circumstances, use of the flow differences would then lead to F-values well below minus one, whereas in this case Eq. (4.2.1) remains between zero and one and was therefore preferred.

The apparent deterioration in the performance of the 3-layer ANN under extremely linear conditions can easily be explained by the inherent non-linearity of the mapping function that is instantiated by an ANN using sigmoid transfer functions. The ANN could be made to reproduce a more linear behaviour by simply replacing the sigmoid function by a linear transfer function, as given in (2.4.25), but this would inevitably result in a decrease in the performance level of the ANN in the presence of even the slightest non-linearity.

In an attempt to further improve the performance of the ANN under extremely non-linear conditions, it was decided to repeat the exercise using 4-layer ANNs (i.e. two hidden layers). The results are summarised also in Figs 4.8 and 4.9, and in Table 4.5. Although Fig. 4.9 indicates a marginal improvement in the level of performance of the ANN for the linear catchment, this is not reflected in the coefficients of efficiency in Table 4.5. The differences between the results of the 3-layer and 4-layer ANNs are, in fact, insignificant in these experiments. The above results together with the increased computational effort required to train a 4-layer ANN compared to that for a 3-layer ANN raises some doubt as to the suitability of the 4-layer ANN for modelling the rainfall-runoff process.

Hertz *et al* (1991, p. 142) also reach the same conclusion. They state that the total number of hidden units necessary to solve a given problem is not generally known and, in fact, this number may grow exponentially with the number of input units. Furthermore, although two or more hidden layers may well permit a solution with fewer units in all, nothing at all can be said about the corresponding learning or generalisation properties. It is possible that some functions are representable but not learnable with two hidden layers, perhaps due to the presence of local minima.

These results tend to support the contention by Rumelhart *et al* (1994) that minimal networks can offer better generalised performance than more complex networks. The extreme accuracy of the ANNs for the typical non-linear case (m = 0.8), which would appear representative of many rainfall-runoff data sets, indicates that a 3-layer network is capable of identifying usable relationships between discharges and antecedent rainfalls for a wide range of catchments with responses varying from the linear to the non-linear, typical of the majority of real-world applications. In terms of individual storm hydrographs, the largest peaks were not always reproduced closely. This performance can be expected when the number of 'high' peaks is small

compared with the number of 'average' peaks in the training data set; the ANN assigns relatively more importance to the latter, rather than to matching the former. These findings are sufficient to suggest that extreme caution would have to be exercised if ANNs were to be employed in studies of extreme floods.

4.5 Extreme Events: The Problem of Extrapolation

As mentioned in §4.4.1, if an ANN were to be applied to a real catchment, even if the training data included all the available measurements, there is always a small but non-negligible probability that an extreme event beyond the range of recorded experience may occur in the future. The RAIN3 data set was therefore generated that contained rainfall maxima outside the range of those upon which the training data was based (see Tables 4.3 and 4.4) The runoff hydrographs produced by applying this rainfall data set to each of the three catchment models were then used as verification data sets on the trained networks from the previous section. The results of these verifications are shown in Fig. 4.10.

The coefficients of efficiency for these experiments were 0.790, 0.848 and 0.775 for the regular, linear and non-linear catchment models respectively. Fig. 4.10 depicts quite clearly the problem of extrapolation that arises when the ANN models are confronted with extreme data that it has not seen before. In each case, the ANN 'cuts-off' the predicted discharges at a value equivalent to the maximum value of discharge in the training data sequences. In these experiments, the maximum discharges in the training data sets were standardised to a value of 0.9 on the sigmoid function output. This in fact allows the network to extrapolate beyond the maximum training discharge by up to 10%, as shown on Fig. 4.10.

The degree of extrapolation exhibited by these networks is by no means sufficient to capture the behaviour of each of the catchment responses. In these experiments, the maximum flows in the verification data sequences have magnitudes that are more than twice the maxima that occur in the training data sequences. Fig. 4.10 shows that this problem of extrapolation is the same for each of the three theoretical catchment models considered.

In fact, the problem of extrapolation is very obviously linked to the choice of the standardisation factors that are applied to the training and verification data sequences. In order to investigate the effect of these standardisation factors in some more detail, further experiments were performed in which other ranges of standardisation were used than the range of 0.1 to 0.9 mentioned in §4.4.3.

Whereas the previous experiments standardised the discharge values at the output node to a range of 0.1 to 0.9, it was decided to set the maximum discharge value in the training set to 0.5 on the sigmoid function. This then keeps up to 50% of the output range of the sigmoid function, i.e. from 0.6 to 1.0, 'free' for possible extrapolation beyond the maximum value of the training data.

For these and subsequent experiments only the data for the regular catchment was used, as this is most indicative of the catchment behaviour that may reasonably be encountered in practice.

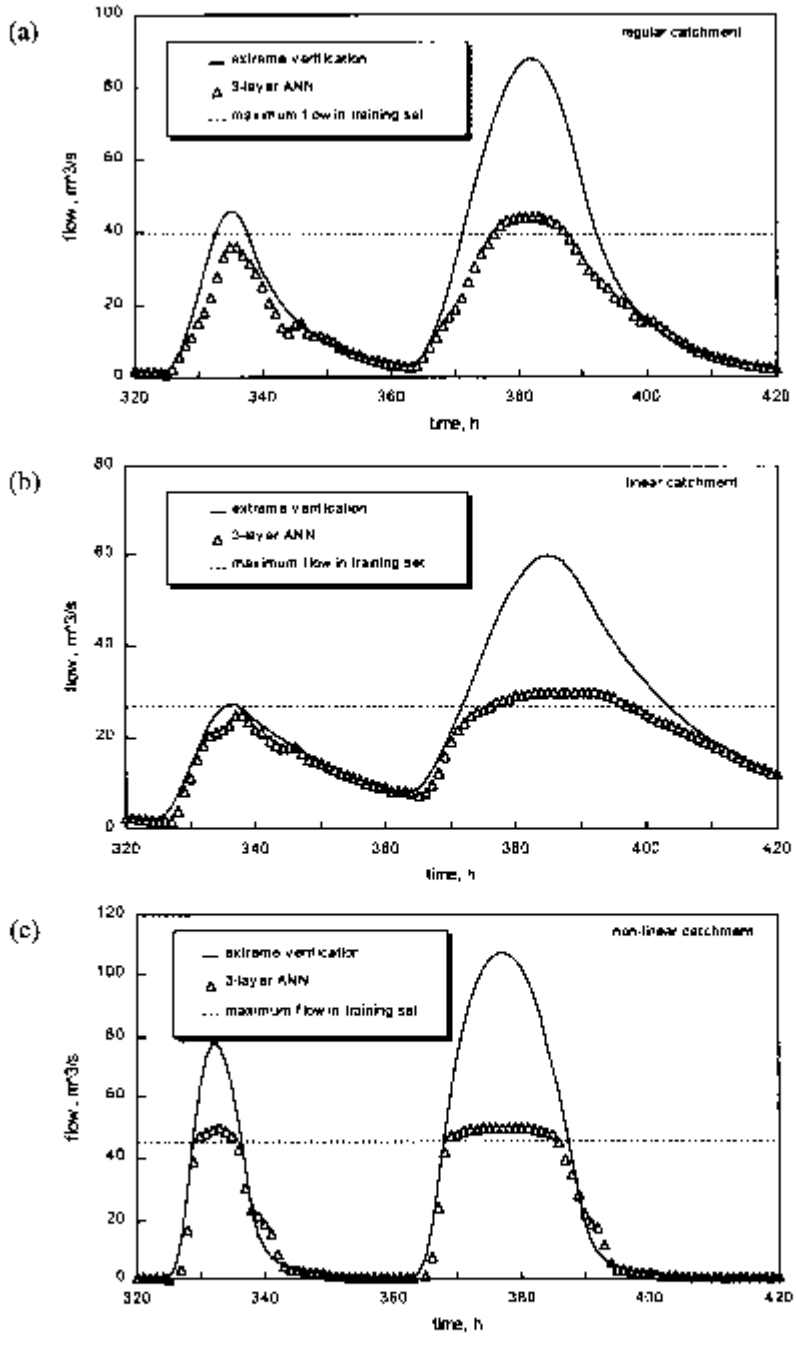


Fig. 4.10 Extreme verification of 3-layer ANNs on input and output from each of the three conceptual catchment models. (a) regular catchment; (b) linear catchment; and (c) extremely non-linear catchment. For clarity of illustration, two events only have been selected

Using this new standardisation range, a 3-layer ANN was trained and verified as before. The performance of this network provided coefficients of efficiency of 0.995 and 0.995 for the training and normal verification sequences respectively, which are comparable to the results obtained previously.

Now, however, when the extreme verification data is applied to the trained ANN, the coefficient of determination for the extreme verification becomes 0.818, compared to 0.790 previously. This is, of course, only a very minor improvement, and is depicted in Fig. 4.11 for the same events shown in Fig. 4.10.

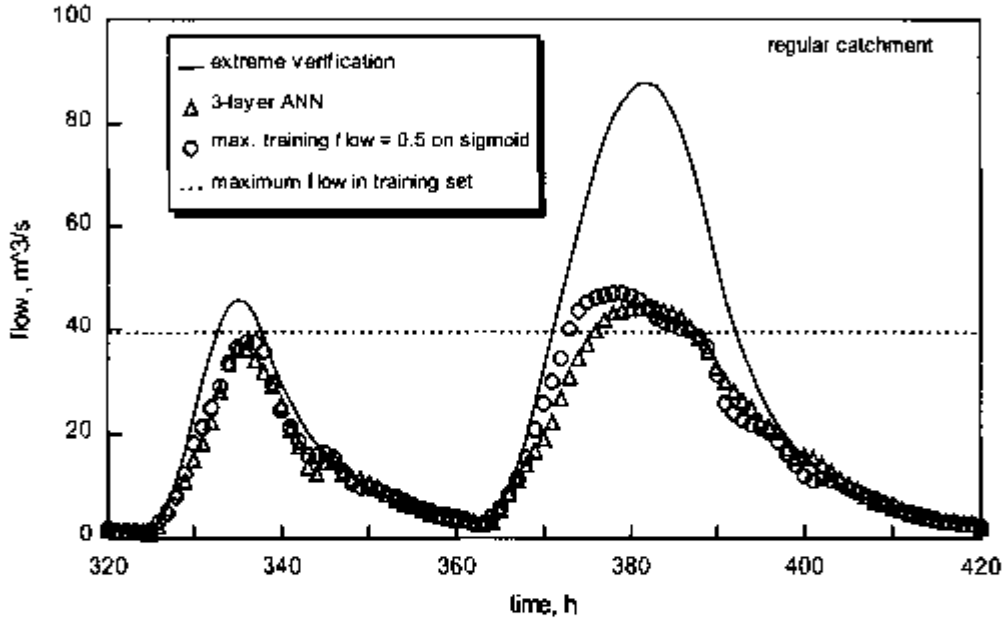


Fig. 4.11 Comparison of the extreme verification results of the 3-layer ANN used previously to a 3-layer ANN in which the training data set was standardised to the range 0.1 to 0.5, thus allowing a degree of extrapolation beyond the maximum value of the training data

The results in Fig. 4.11 show that this approach is not very satisfactory for solving the extrapolation problem. Indeed, the degree to which the ANN extrapolates beyond the maximum training value has only increased from about 10% to 20%, even though the output neuron has a theoretical extrapolation range of up to 100% of the maximum training value. This is quite obviously a problem of 'saturation' of the neurons in the hidden layer, caused by the associated extreme values of the input rainfalls and antecedent flows. The input data sequences contain values with magnitudes that are about twice the training values, resulting in weighted sums to the neurons in the hidden layer that are either very large or very negative, so that the outputs from the neurons are 'stuck' at either zero or 1.0.

Another approach to the choice of standardisation values is to consider the actual range of values in the verification data sequences. For example, the extreme verification data for the regular catchment model has a maximum rainfall ordinate of 13.0 mm/h and a maximum discharge

ordinate of 87.95 m^3/s. Using the standardisation factors based on the training data set, this leads to a maximum standardised rainfall value of 7.5 instead of 3.0 at the input nodes, and a maximum standardised flow value of 2.0 at the output node. The standardised output value of 2.0 is, of course, impossible to represent on the sigmoid transfer function, which is limited to a maximum value of 1.0. It is possible, however, to 're-standardise' the extreme verification data sequence so that the maximum standardised rainfall value is 3.0 and the maximum standardised flow value is 0.9. This was done, and this re-standardised extreme verification sequence was simply applied to the original trained 3-layer ANN. The results for the two selected verification events are depicted in Fig. 4.12.

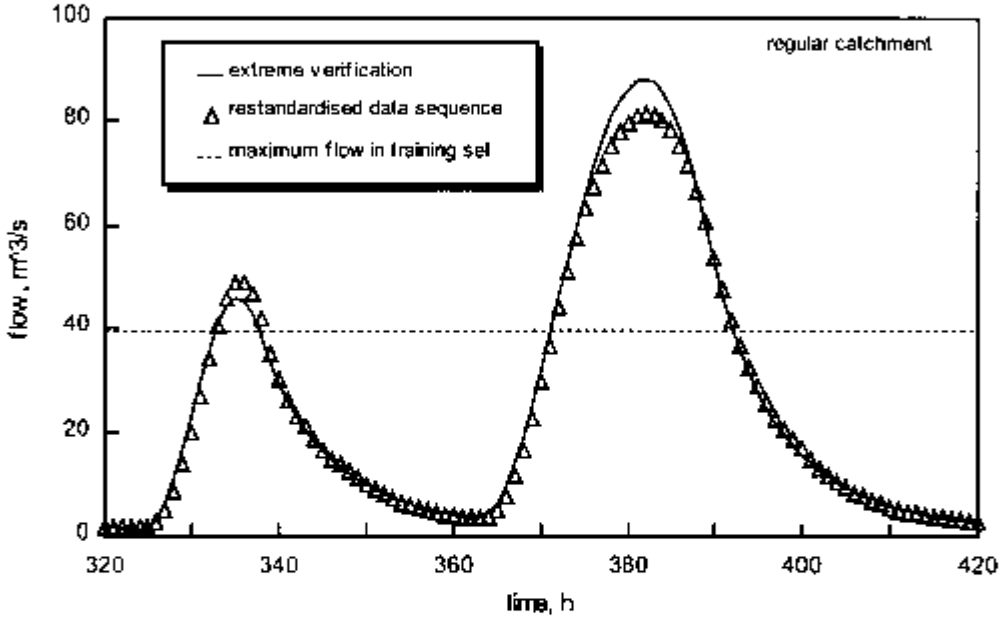


Fig. 4.12 Extreme verification of a 3-layer ANN in which the verification data sequence is standardised using its own maxima and not the maxima of the training data sequence

The coefficient of efficiency for the entire restandardised verification sequence has now increased to 0.995, which is comparable to the coefficients of efficiency obtained for the training and 'normal' verification data sequences (Table 4.5). Although this outcome appears to be quite promising, considerable care should be exercised in interpreting these results. By restandardising the extreme verification data sequence we are actually reassigning a different 'meaning' to the input and output values than the one that was assigned during the training process. For example, although we believe that the maximum rainfall ordinate is 13.0 mm, corresponding to a standardised value of 3.0, the ANN only interprets this to be 5.2 mm. Similarly, the ANN can only output a maximum discharge value of 43.7 m^3/s, corresponding to a standardised value of 1.0, but we now interpret this to be 97.7 m^3/s. The ANN is in fact modelling a regular catchment under normal conditions, and we are simply taking these results and rescaling them by a factor of between 2.2 and 2.5 to obtain results under extreme conditions.

The accuracy of the results obtained indicates that the ANN has indeed captured quite well the essential rainfall-runoff relationship for this type of catchment. Furthermore, if the rainfall-runoff process was an entirely linear process, i.e. twice the rainfall were to lead to twice the runoff, then rescaling the inputs and outputs for extreme events would be quite acceptable. The rainfall-runoff process is, however, not at all linear, so that the above rescaling is, in this case, quite meaningless, and even rather dangerous to apply. Obviously, for the theoretical catchment used above, there is a large degree of linearity arising due to the absence of non-linear hydrological parameters, like infiltration, evaporation, etc., that have been excluded from the RORB model used here. Further testing of the ANN models using real catchment data is therefore required.

4.6 Silk Stream and Dollis Brook Catchments

Minns and Hall (1997) demonstrate the ability of an ANN to learn a general rainfall-runoff relationship applicable to a hydrologically-homogeneous area. In order to investigate whether the relationships between rainfall and runoff learned by an ANN had any hydrological significance. networks were fitted to data from two drainage areas situated to the north of London. and previously analysed by Hall (1977). The Dollis Brook and the Silk Stream are two adjacent catchments of 23.99 and 31.25 km2 draining into the Brent Reservoir. The geology of both areas is predominantly London Clay, with outcrops of the Claygate beds and the Pebble beds occurring on the higher ground to the north. Since the 1920s, both catchments have been extensively urbanised, such that the proportions of impervious area had by 1970 reached 21 and 25 per cent for the Dollis Brook and the Silk Stream respectively.

Records of stream flows were available for Dollis Brook at Hendon Lane for 40 events over the period 1952-69, and for the Silk Stream at Colindeep Lane for 51 events over the years 1929-44. Individual storms were abstracted and paired with rainfall data from autographic raingauges at Stanmore and Edgeware (Silk Stream) and Barnet and Mill Hill (Dollis Brook). A 30-minute time interval was used throughout. With these periods of record, the amount of impervious area in the Dollis Brook increased by only 3 per cent, whereas that of the Silk Stream increased from 14 to 21 per cent.

The analyses began by dividing the available storms into training and verification data sets. For the Dollis Brook, the first 23 events were used for training, and the last 17 for verification. This division ensured that the maximum flow peak of 16.8 m^3/s was included in the training data. Since the Silk Stream had been subjected to a significant increase in urban development during the period of the record, the first 13 and the last 10 storms were included in the training data set, and the remaining 28 were reserved for verification. This mixture of storms not only conveniently encapsulated the historical changes in percentage impervious area, but also ensured that the maximum flow in the training data of 9.8 m^3/s was smaller than the peak of 13.8 m^3/s in the verification data. Both features were introduced for the purposes of the numerical experiments reported below.

Although individual events were extracted, the data were presented to the networks in the form of time series, i.e. each event was placed in sequence, with a clear period of hydrograph recession being included prior to the start of the succeeding event. Sudden jumps in flow magnitude were

avoided, even at the expense of inserting a short, artificial extension of the previous recession, in order to avoid the introduction of false features that could be learned as readily as the 'true' catchment behaviour.

The ANN models for both catchments were constructed with eight inputs (rainfall at time t and the four previous time steps, and the flows at three previous time steps) and one output of flow at time t. This selection was based upon a series of initial trials with different combinations of rainfall and flow inputs, and was partly constrained by the need to use the same configuration for both catchments, as will become more apparent in the discussion below. In addition, further trials with different numbers of nodes in the hidden layer indicated that the use of three or four hidden nodes provided the smallest network configurations with the best results on both the training and on the verification data sets as shown in Table 4.6 (see also Appendix 1). For the following experiments, four hidden nodes were subsequently used.

Table 4.6 RMS errors calculated by the WinNN software for various configurations of ANN for both training and verification data sequences

Number of hidden nodes	RMS errors	
	training data set	verification data set
2	0.0082	0.00095
3	0.0077	0.00098
4	0.0076	0.00081
5	0.0074	0.00127

An ANN for each catchment was trained and then verified on the data sets compiled as described above. The results are summarised in Table 4.7, and show that for both catchments efficiencies in excess of 97 per cent were achieved in training, a result which the majority of hydrological modellers would be gratified to achieve! For the verification events, the coefficient of efficiency of the Dollis Brook network is slightly improved but that of the Silk Stream network falls from 0.974 to 0.949.

Table 4.7 Coefficients of efficiency for the training and verification of neural network models of the Silk Stream and Dollis Brook catchments using five rainfalls and three flows as input and one flow as output

Catchment	ANN model	Training data	Verification data
Dollis Brook	Dollis Brook	0.979	0.983
Silk Stream	Silk Stream	0.974	0.949
Silk Stream	Dollis Brook	0.920	0.957

The reason for the difference in performance between the two ANN models is readily apparent when the plots of simulated and observed flows are examined in Figs 4.13 and 4.14. Rather than present the whole verification data set for each catchment, which is in excess of 2000 points, a set of 300 ordinates has been selected to illustrate the problem.

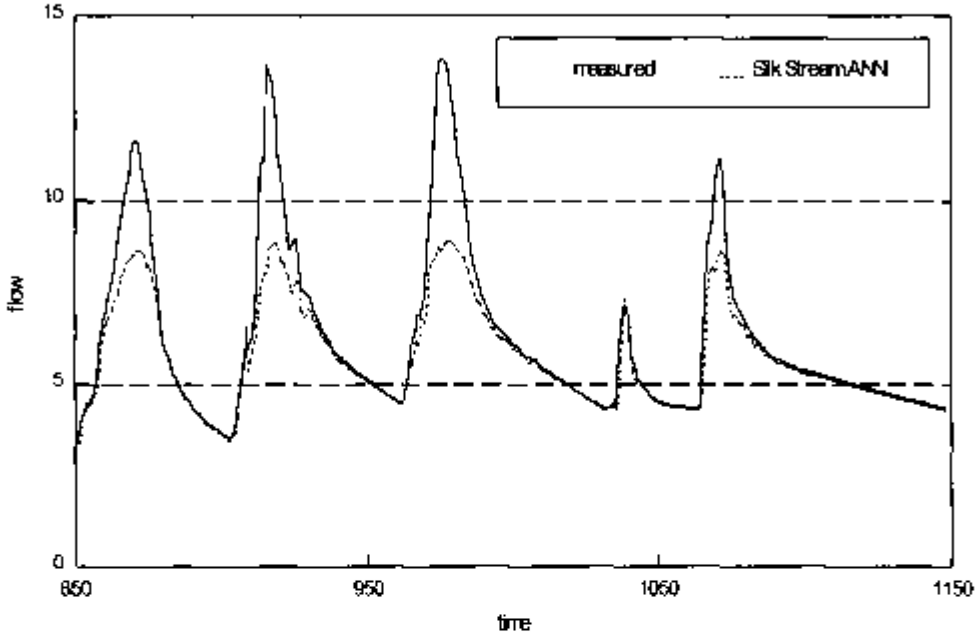


Fig. 4.13 Segment of 300 ordinates of the verification data set for the Silk Stream using an ANN with five rainfalls and three flows as input and one flow as output

In Fig. 4.13 for the Silk Stream catchment, the higher peaks in the verification data set, which exceed those in the training data set in magnitude, are not captured correctly. By way of contrast, the peaks in the training data set for the Dollis Brook catchment were higher than those in the verification data set, resulting in a much more flattering performance level for this catchment, as shown in Fig. 4.14.

These results clearly demonstrate that, if sufficient care is exercised in the selection of the training and verification data sequences, then many problems of extrapolation might be avoided in practice. For catchments with sufficiently long historical records, the training data sequence must include the maximum recorded historical events so that the ANN can learn to reproduce the response of the catchment over the widest possible range of hydrological and meteorological conditions, as was demonstrated in the Dollis Brook catchment model above. This will not, of course, preclude the necessity for extrapolation during some unforeseen future event that is greater than all historical events. There will always be a small, but non-negligible, probability that some maximum probable rainfall or discharge event could occur in the future. As was mentioned above, the Silk Stream training and verification data sequences were in fact specifically chosen to highlight this problem of extrapolation on real catchment data for the purposes of the following investigations.

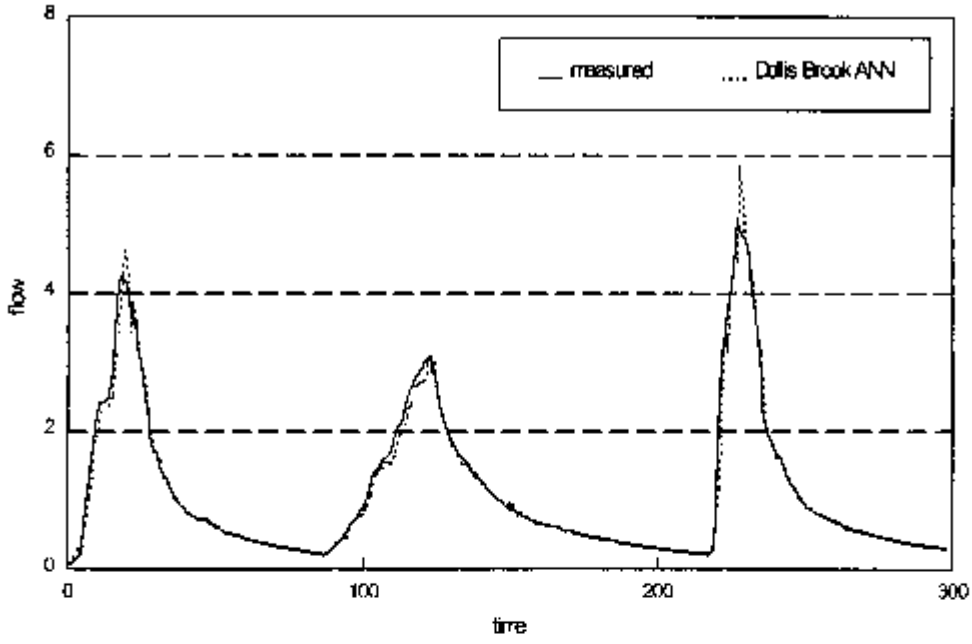


Fig. 4.14 Segment of 300 ordinates of the verification data set for the Dollis Brook using an ANN with five rainfalls and three flows as input and one flow as output

4.6.1 Transferability of the network

The training and verification data for the Silk Stream catchment were now presented to the ANN that had previously been trained on the Dollis Brook data. These results are also summarised in Table 4.7. Since the Dollis Brook ANN had been trained on data containing a maximum flow of 16.8 m^3/s, as opposed to the 9.8 m^3/s of the Silk Stream training data, the former actually performed better on the Silk Stream verification data (maximum flow 13.8 m^3/s) than did the Silk Stream ANN itself ! Fig. 4.15 shows the plot of the same 300 ordinates of the Silk Stream verification data as used in Fig. 4.13, but now the output has been produced by the ANN trained on the Dollis Brook data; the so-called Dollis Brook ANN. The peak flows are now reasonably well caught, but the recessions are tending to undershoot and there is some evidence of 'noise'.

These results provide some indication that the ANNs of these two adjacent catchments, or for that matter, any other hydrologically-similar catchments, are interchangeable, but that performance is still confounded by the range of standardisation of the output data. The possibilities of using ANNs developed from data for one catchment as models for the response of other, hydrologically-similar areas are inextricably linked with the problems of extrapolation. Nevertheless, the superior performance of the Dollis Brook ANN on the Silk Stream data appears to indicate that some interchangeability is possible, perhaps even to the extent of developing an ANN for an 'ungauged catchment'. However, in this case, an extended ANN model would be

required to incorporate factors that would reflect different land uses and even the changes in those land uses over time. This approach is investigated in more detail below.

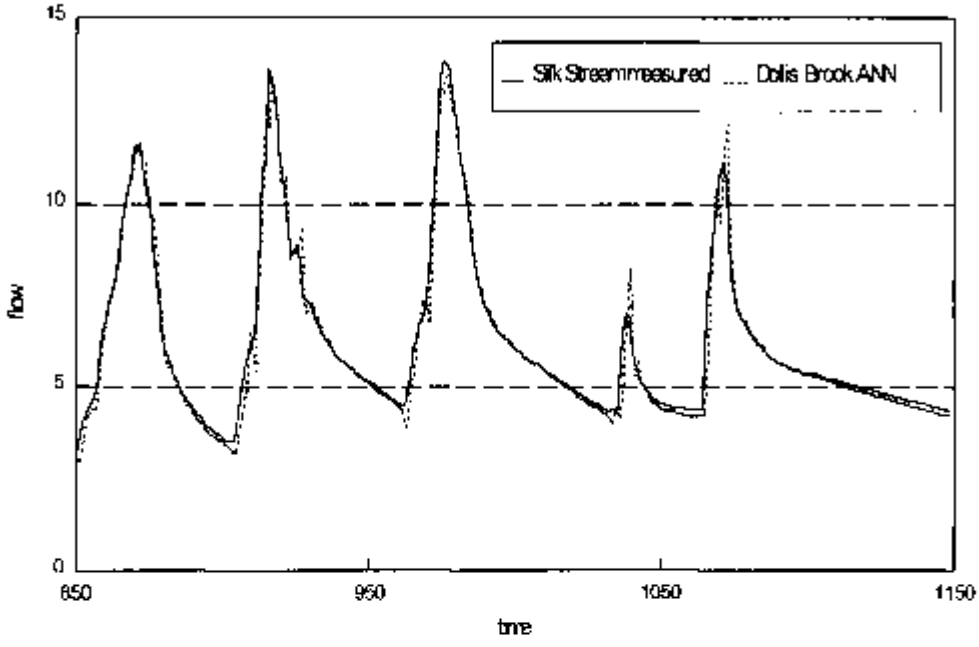


Fig. 4.15 Segment of 300 ordinates of the verification data set for the Silk Stream using an ANN derived for the adjacent catchment of the Dollis Brook

4.7 The Extrapolation Problem Revisited

In an attempt to improve the performance of the Silk Stream ANN, the network was retrained using the *change in discharge*, ΔQ, as the network output rather than the absolute flow ordinate. This choice has its origins in the widely-used empirical rule, dating back at least to F.F. Snyder in the late 1930s (*see* Johnstone and Cross, 1949) and promulgated in the UK Flood Studies Report (Natural Environment Research Council, 1975), that at least 5 or 6 points should be used to define the rising limb of a finite-period unit hydrograph. In effect, this rule is an attempt to restrict $\Delta Q/\Delta t$, and therefore ΔQ. Furthermore, when anterior values of discharge are introduced as inputs to the ANN, then the resulting discharge output from the ANN can better be regarded as a model of the *change* in discharge from one time step to another rather than a prediction of the absolute value of the discharge.

An ANN was now trained and verified using exactly the same data as that which was used to train the ANNs in §4.6.1, but now incorporating an output of ΔQ at time t rather than Q at time t. The coefficients of efficiency for the training and verification of the Dollis Brook ANN show little change over the previous experiments, as summarised in Table 4.8. The differences in the

behaviour of the Silk Stream ANN are, however, more obvious. In training, the coefficient of efficiency is almost the same, but in verification, the coefficient of efficiency has increased from 0.949 to 0.980.

Table 4.8 Coefficients of efficiency for the training and verification of neural network models of the Silk Stream and Dollis Brook catchments using five rainfalls and three flows as input and the change in flow, ΔQ, as output

Catchment	ANN model	Training data	Verification data
Dollis Brook	Dollis Brook	0.973	0.983
Silk Stream	Silk Stream	0.975	0.980
Silk Stream	Silk Stream with IMP	0.970	0.982

These results appear to indicate that the use of first differences in flow effects some improvement in performance of the networks, especially under extrapolation conditions. However, as shown in Fig. 4.16, the larger peaks in the verification data set are now overestimated and the output is somewhat noisy.

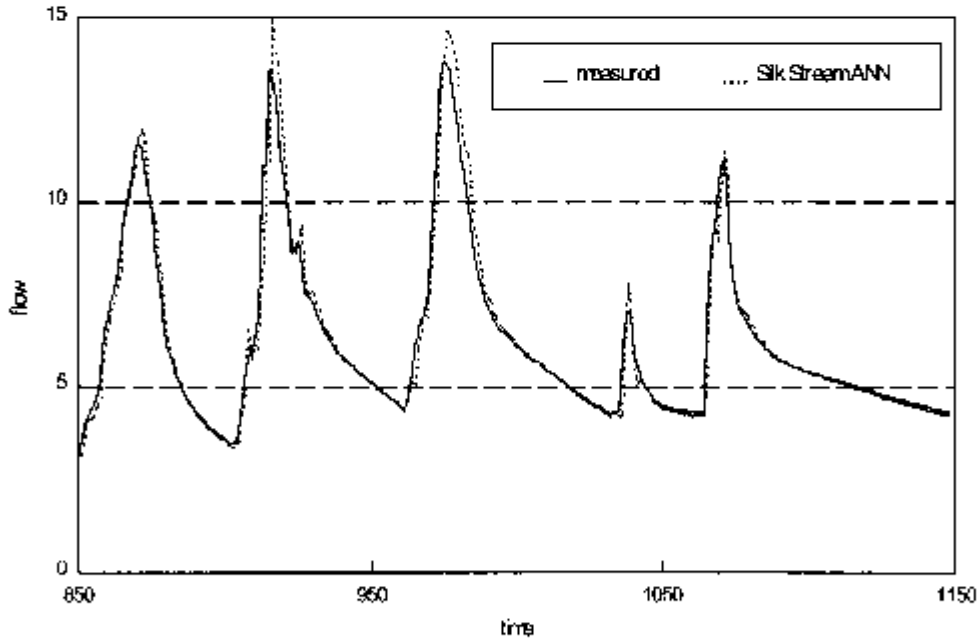


Fig. 4.16 Segment of 300 ordinates of the verification data set for the Silk Stream using an ANN with five rainfalls and three flows as input and the change in flow, ΔQ, as output

The overshooting in the results suggests that the subtlety of the hydrological response has not been fully caught. This result is not altogether unexpected, since the Silk Stream has been

subjected to urbanisation. As mentioned before, the amount of impervious area in the Silk Stream catchment increased from 14 to 21% during the period of data measurements. The possibility of incorporating some index of change led to an extension to the above model with an additional input data stream consisting of the percentage impervious area, IMP, applicable to the catchment for the year in which the storm event was recorded. A new ANN was configured and trained that now consisted of 9 input nodes (i.e. 5 rainfalls, 3 discharges and the percentage impervious area) and one output node for the discharge. For this particular example, the generalisation properties of the network were also considerably improved when the number of hidden nodes was reduced from four to three.

The results for both training and verification are also summarised in Table 4.8, and the visual fit of the same 300 ordinates of the verification data are shown in Fig. 4.17.

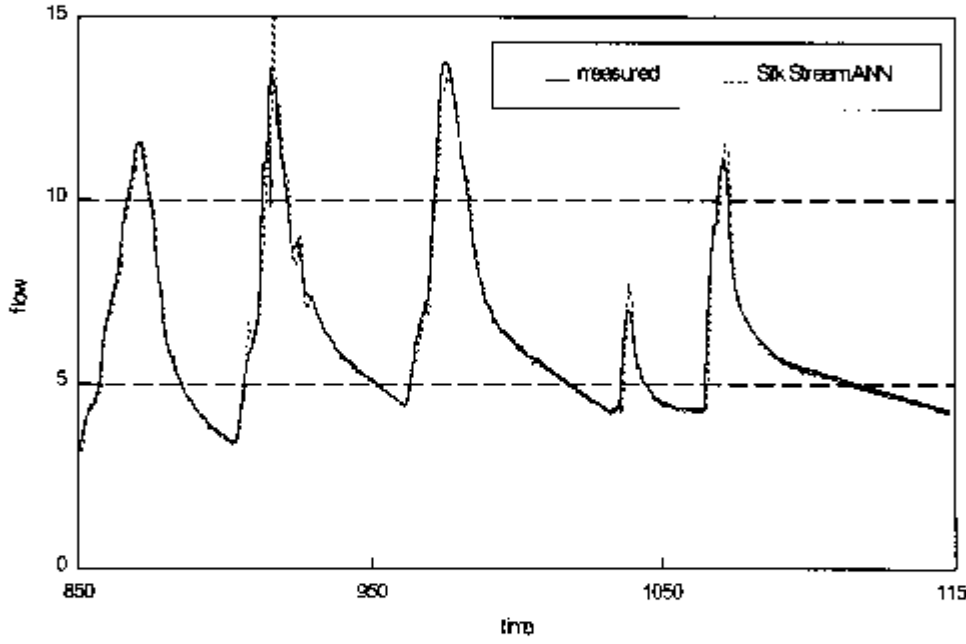


Fig. 4.17 Segment of 300 ordinates of the verification data set for the Silk Stream using an ANN with five rainfalls, three flows and the percentage impervious area, IMP, as input and the change in flow, ΔQ, as output

The very slight decrease in the coefficient of efficiency given in Table 4.8 for the training data sequence can be explained by the fact that the input data array is now more complex, containing more information than before, making the relationships slightly more difficult to learn. There is also one less node in the hidden layer compared to the previous network, which reduces the number of degrees-of-freedom that the ANN has in fitting a relationship to the training data. The reason for this change was the observed tendency of this ANN to *overlearn* and subsequently overfit the results. This was overcome by reducing the hidden nodes in the network and by stopping the training at the moment that the overfitting started (see Appendix 1). For the verification data sequence, the coefficient of efficiency has only improved in the third decimal place, but the peaks and recessions appear now to be much better fitted.

These results demonstrate that the inclusion of an extra input, consisting of the percentage of impervious area, improves the visual fit of those flood peaks that are not included in the training data set for an urbanising catchment. However, the noisiness of the ANN output, particularly in the second event of Fig. 4.17, appears to indicate that there may be even further scope for refinement of this model. The above experiments indicate the direction in which further progress in applying artificial neural networks might proceed.

In summarising then, the results of all of the numerical experiments reported above indicate that suitably configured artificial neural networks are capable of identifying usable relationships between runoff discharges and antecedent rainfall depths to an exceptional degree of accuracy. The relationships are obtained using only the raw, measured data and do not require the use of any derived or artificial calibration parameters.

In particular, attention should be drawn to the fact that the ANN model provides these exceptional results unhindered by constraints of volume continuity in the input and output data and, in fact, the units of the data are chosen simply for convenience of measurement and representation (e.g. rainfall depths in mm, discharges in m^3/s). Furthermore, simple, non hydrological parameters like the percentage of impervious area may be easily incorporated into the model at the discretion of the modeller. These types of parameters may be derived from simple measurements or may even be highly intuitive, and are likewise unrestricted in terms of conditions of dimension or hydrological-physical consistency.

5 Modelling of Pure Advection Processes

5.1 The Scalar Wave Equation

It is easily shown (e.g. Abbott, 1979/1992, p. 102; Abbott and Minns, 1997, p. 162) that a wide class of rainfall-runoff models of the kind described in the preceding chapter can be described in terms of pure advection processes (see also the results of Babović and Abbott, 1997). This is so because the pure advection, or scalar wave equation:

$$\frac{\partial z}{\partial t} + c(z)\,\frac{\partial z}{\partial x} = 0 \tag{5.1.1}$$

describes the movement or 'transport' of any property of the fluid, z, with a representative velocity or 'celerity', c, which may even be a function of the fluid property itself, i.e. $c = c(z)$. The partial differential equation (5.1.1) lends itself very conveniently to solution by traditional numerical methods, such as those referred to as finite-difference methods (Abbott and Minns, 1997, pp. 196 *et seq*), but, if not most carefully constructed, these methods may suffer tremendously from problems of accuracy and stability, especially when the local celerity is not constant in time and space.

For example, the simplest finite difference scheme, which consists of a forward difference approximation in time and a backward difference approximation in space, produces a translation of the differential equation (5.1.1) into a finite difference equation of the form:

$$z_j^{n+1} = (1 - Cr)\, z_j^n + Cr\, z_{j-1}^n \tag{5.1.2}$$

where

$$Cr = \text{Courant Number} = \frac{c\Delta t}{\Delta x} \tag{5.1.3}$$

and $j-1, j, j+1$ represent adjacent gridpoints in space, separated by Δx; and $n, n+1$ represent consecutive time levels, separated by Δt.

The finite difference scheme (5.1.2) produces an exact solution to the differential equation (5.1.1) only when $Cr = 0.0$ and $Cr = 1.0$, in which case the solution reads:

$$z_j^{n+1} = z_j^n \tag{5.1.4a}$$

and

$$z_j^{n+1} = z_{j-1}^n \tag{5.1.4b}$$

respectively.

If $Cr > 1.0$, the solution of (5.1.2) produces unstable behaviour and the results are meaningless. On the other hand, if $0 < Cr < 1.0$, the solution of (5.1.2) suffers from rather severe numerical diffusion. A Taylor series expansion of (5.1.2) shows that the finite difference equation (5.1.2) is in fact not exactly equivalent to the differential equation (5.1.1) under all conditions. It can be shown that (5.1.2) corresponds to the differential equation:

$$\frac{\partial z}{\partial t} + c\,\frac{\partial z}{\partial x} = \frac{1}{2}\,c\,\Delta x\,(1 - Cr)\,\frac{\partial^2 z}{\partial x^2} + \text{higher-order terms} \tag{5.1.5}$$

where we can see that (5.1.5) differs from (5.1.1) only by an 'amount' represented by the right-hand side of (5.1.5). We call the expression on the right-hand side of the equality of (5.1.5) the *truncation error* of the finite difference scheme (5.1.2) relative to the differential equation (5.1.1) (*see*, for more detailed derivations, Abbott and Minns, 1997, p.202). The most significant terms in the truncation error expression in (5.1.5) have been combined into a single, second-order space-derivative term that clearly demonstrates the diffusive nature of this term. It is obvious from (5.1.5) that the numerical diffusion will be at a minimum when the truncation error is at a minimum. The most significant term of the truncation error is equal to zero when the celerity, $c = 0$ and hence $Cr = 0.0$, as well as with $Cr = 1.0$. In these cases, the solution of (5.1.2) reduces exactly to (5.1.4), so that the truncation error has in fact disappeared completely.

The pure advection equation arises in many applications in hydraulics and hydrology. The *kinematic wave approximation* used in river hydraulics is simply another form of the pure advection equation (Abbott and Minns, 1997, p. 71). It has been shown by Cunge (1969) that the well-known Muskingum Method for flood wave propagation can actually be represented by an approximate solution of the pure advection equation, where the desired flood wave attenuation is obtained artificially through the numerical diffusion. Furthermore, as already introduced, it can be shown that the *instantaneous unit hydrograph method* of rainfall-runoff modelling can be represented as a linear system of pure transport (Abbott and Minns, 1997, p. 162). The search for a stable and accurate solution technique for problems of pure advection is still the subject of much ongoing research in numerical modelling (*see, for example,* Cunge *et al*, 1980; and Verwey and Ilić, 1993). This chapter describes an investigation into the application of artificial neural network to the problem of pure advection modelling.

5.2 The ANN as an Explicit Numerical Scheme

The first experiments in this investigation involved training an ANN to reproduce the exact solution of the pure advection equation on a finite grid. The input array to the ANN consisted of two values representing the value of the function z at the gridpoints j and $j - 1$ at time level n, while the output consisted of the value of the function z at gridpoint j at time level $n + 1$. The patterns used for training and verification of the ANN consisted of a simple triangular distribution and a Gaussian distribution of the function z along the x-axis, as illustrated in Figs 5.1 and 5.2 respectively.

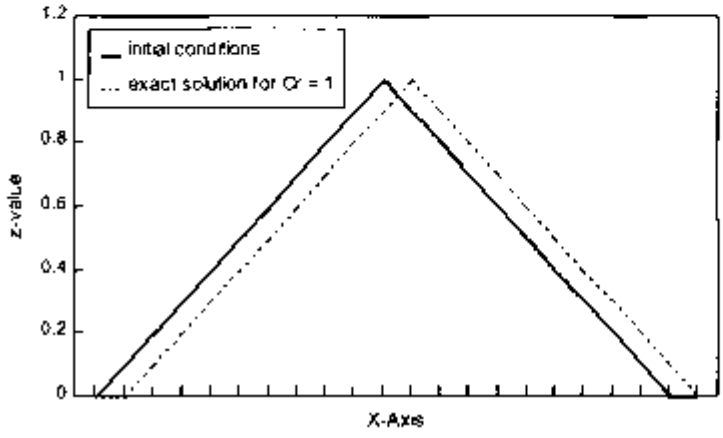


Fig. 5.1 Triangular distribution of a function, z, and the exact solution to the problem of pure advection one time-step later with Cr = 1

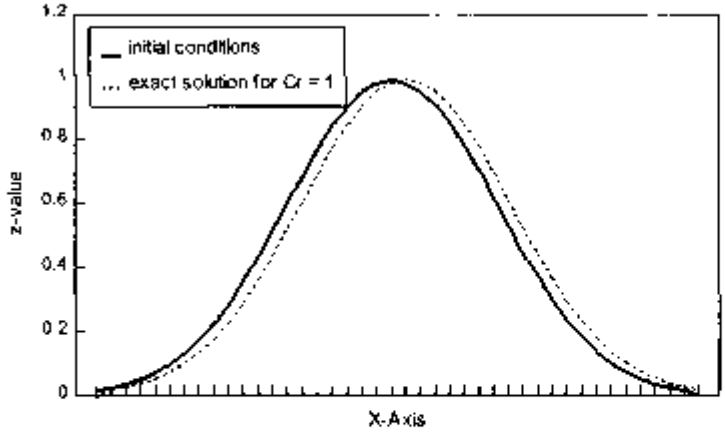


Fig. 5.2 Gaussian distribution of a function, z, and the exact solution to the problem of pure advection one time-step later with Cr = 1

For $Cr = 1$, the exact solution of the pure advection equation means that the distributions should be translated exactly one gridpoint in the x-direction during one time step, which is also illustrated in Figs 5.1 and 5.2.

As this is a linear process, it is obvious that the simplest linear perceptron should suffice for this problem, and indeed may be the most eminently suitable. If the output from the artificial neurons is taken to be linear, that is:

$$f\left(\sum_i w_{i,j}\, x_{i,\lambda}\right) = \sum_i w_{i,j}\, x_{i,\lambda} \tag{5.2.1}$$

then the simplest network that would solve this problem is a two-layer network (i.e. no hidden layers) as shown in Fig. 5.3. Note that this simple network is in fact identical to the Rosenblatt perceptron that was discussed in §2.3 but now using the linear activation function (2.4.25) instead of a Heaviside step function.

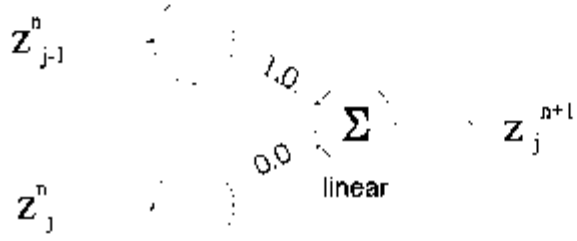


Fig. 5.3 The simplest possible two layer network (perceptron) to solve the pure advection equation for Cr = 1

The weight distribution shown in Fig. 5.3 is in fact the exact configuration that was obtained when a 2-layer, linear ANN was trained using the data for the triangular distribution of z, that was sketched in Fig. 5.1. Due to the fact that the activation function used in the above example is linear, we can derive the mathematical expression of the solution in Fig. 5.3 as a simple weighted sum of the input variables, i.e.:

$$z_j^{n+1} = 0.0 \times z_j^n + 1.0 \times z_{j-1}^n$$

which is identical to the exact solution given earlier as (5.1.4b).

One hidden layer was then introduced into the above network with linear threshold functions maintained throughout. This ANN was then trained on the same data that was used above (Fig. 5.1). For the training of this ANN, the back-propagation algorithm converged extremely quickly to a residual RMS error (i.e. the error between the desired outputs and the outputs produced by the ANN) of less than 10^{-6}. The final distribution of weights is shown in Fig. 5.4.

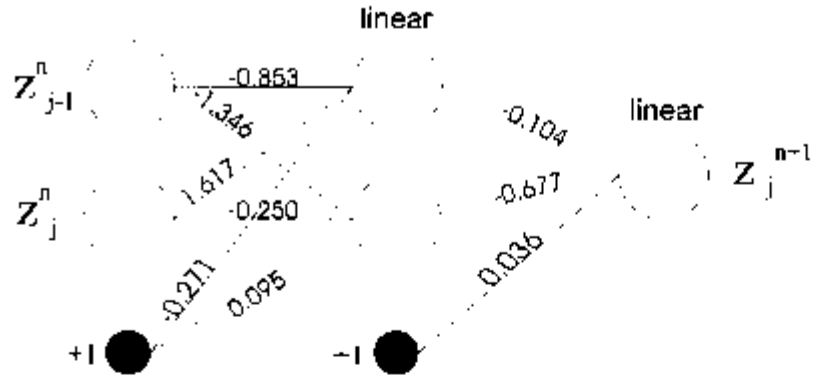


Fig 5.4 Configuration of weights for a 3-layer, linear ANN with two inputs that has been trained to reproduce the pure advection equation for Cr = 1

The smaller shaded neurons in Fig. 5.4 represent the so-called bias neurons that always have an input signal of +1 and have the effect of off-setting the threshold function, as described earlier in §2.4. The weights attached to the bias neurons are learned during the training process along with the other connection weights

The configuration of the ANN in Fig. 5.4 can be expressed mathematically as:

$$z_j^{n-1} = -0.104\left[-0.853 z_{j-1}^n + 1.617 z_j^n - 0.271\right] \quad \text{(first hidden node)}$$

$$0.677\left[1.346 z_{j-1}^n - 0.250 z_j^n + 0.095\right] \quad \text{(second hidden node)}$$

$$+0.036\left[+1\right] \quad \text{(bias node in hidden layer)}$$

which reduces to:

$$z_j^{n-1} = 1.000 z_{j-1}^n + 0.000 z_j^n + 0.000 \qquad (5.2.2)$$

(5.2.2) is, once again, exactly equivalent to the exact solution of the problem given by (5.1.4b). The above experiments indicate that a linear ANN is indeed capable of learning the exact linear relationship represented by the solution of the pure advection equation under the condition that $Cr = 1.0$.

It is, however, much more common to use sigmoid threshold functions in multi-layer perceptron networks. In order to see how such a non-linear network could cope with this purely linear problem, the ANN used above was subsequently retrained on the triangular distribution of z, using sigmoid threshold functions throughout. The most noticeable change in performance of this

non-linear ANN compared to the linear ANN was the marked increase in time required to train the network. Additionally, the residual error, which is calculated as the mean square error between the desired output values and the output values produced by the ANN, was significantly higher than the residual error obtained with the linear ANN. The final configuration of the weights that was obtained is shown in Fig. 5.5.

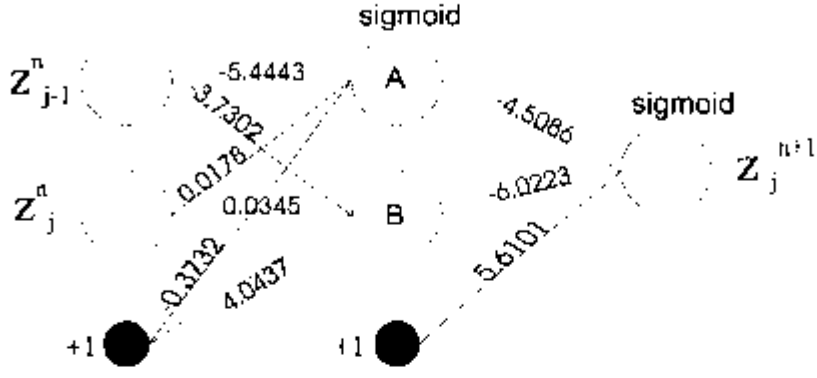


Fig 5.5 Configuration of weights for a 3-layer, non-linear ANN (i.e. sigmoid threshold functions) with two inputs that has been trained to reproduce the pure advection equation for Cr = 1

The most obvious feature of the distribution of weights shown in Fig. 5.5 is the relative magnitude of the weights attached to the z_j^n node in the input layer. These weights are up to 100 times smaller than all other weights in the network. The influence of the z_j^n value upon the final value of the output is therefore almost negligible. This indicates that the ANN has learned to ignore the z_j^n value when calculating z_j^{n+1}, which is an entirely correct conclusion based upon our knowledge of the nature of the exact solution (5.1.4). If we now in fact neglect the effect of z_j^n, the resulting mathematical expression derived from the weight distribution of Fig. 5.5 is:

$$z_j^{n+1} = \frac{1}{1 + e^{-[-4.5086\, o_A - 6.0223\, o_B + 5.6101]}} \qquad (5.2.3)$$

where

$$o_A = \frac{1}{1 + e^{-[-5.4443\, z_{j-1}^n - 0.3732]}}$$ = the output from hidden node *A*;

and

$$o_B = \frac{1}{1 + e^{-[-3.7302\, z_{j-1}^n + 4.0437]}}$$ – the output from hidden node *B*.

It is quite obvious that the exact mathematical solution given in (5.1.4b), that is $z_j^{n+1} = z_{j-1}^n$, cannot be obtained from (5.2.3). This solution can, therefore, only be an approximation of the exact solution. Fig. 5.6 illustrates the accuracy of this approximation by plotting the verification results that were obtained when the trained ANN was subsequently applied to the problem of the advection of a Gaussian distribution for three time steps with $Cr = 1$.

For the first time step, the initial distribution, which is sketched in Fig. 5.2, was used to produce the input data array, i.e. $[z_{j-1}^n , z_j^n]$, for each calculation point in the solution domain. The ANN is then applied at each gridpoint to produce an array of $[z_j^{n+1}]$, which thus constitutes the solution after one time step. The results for the second and consecutive time steps were obtained by using the output array of the previous time step calculation to produce the input data array for the subsequent calculation. In this way, the residual errors that are generated at each time level are also propagated through the solution in each consecutive calculation. The effect of the propagation of the residual errors is quite clearly illustrated in Fig. 5.6.

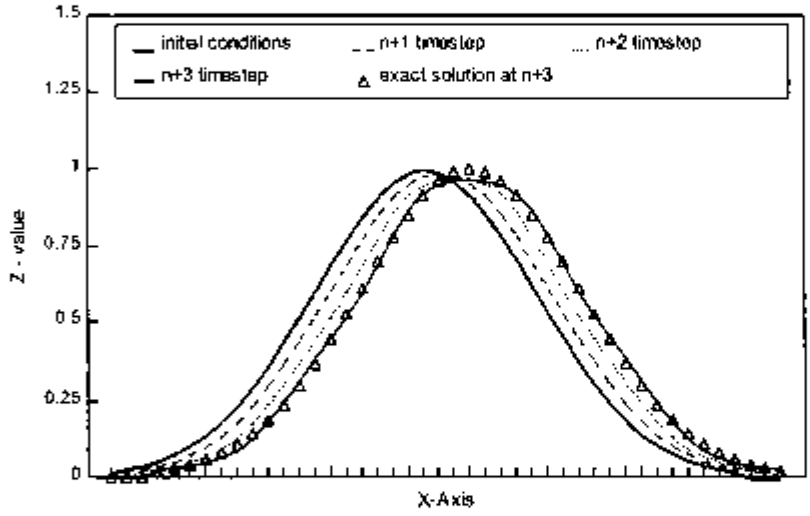


Fig. 5.6 Verification results obtained by applying a non-linear ANN (i.e. sigmoid threshold functions) to the problem of pure advection of a Gaussian distribution for three time steps, n+1, n+2 and n+3, using Cr = 1

Fig. 5.6 indicates that the ANN has learned how to translate the given initial conditions through a distance of one gridpoint per time step, corresponding to $Cr = 1$. The residual error in the training, however, has resulted in a slight underestimation of the peak value of z at each time step, which, of course, gets progressively worse as the errors are accumulated at each consecutive time step. This underestimation has led to a 'loss of mass' in the distribution of z, expressed as the area under each of the solution curves, and which, in this case, has decreased from a level of 17.66 in the initial distribution to a level of 17.49 after three time steps, a loss of almost 1%. Although this solution is still relatively accurate, it cannot compare at all to the accuracy of the exact solution, $z_j^{n+1} = z_{j-1}^n$, which was obtained using the linear ANN and which, naturally, has exactly zero error.

As has been demonstrated above, the solution to the pure advection equation for $Cr = 1$ is almost trivial. A more challenging problem to be considered is what happens when the Courant number is not equal to 1. The major problem with numerical schemes that are used to solve the pure advection equation is their rapid deterioration in performance under conditions when the Courant number varies. The following experiment was therefore carried out to determine if a suitable configuration of an ANN could be developed and trained to perform the required numerical operations under a variety of Courant numbers. The data used for training the network was derived from the Gaussian distribution that was sketched in Fig. 5.2 and the data used for verification of the trained ANN was derived from the triangular distribution that was sketched in Fig. 5.1.

From our knowledge about the nature of the solution on a finite grid, we know that the exact solution to the pure advection equation can only exist for integer values of Cr, which coincide with the exact translation of the solution values from one gridpoint to the other. It was therefore proposed to train an ANN using the exact known solutions for three different Courant numbers, for example $Cr = 0.0$, 1.0 and 2.0. The training data sequence therefore consisted of an input array: $[z_{j-2}^{n}, z_{j-1}^{n}, z_{j}^{n}, Cr]$, and a desired output array: $[z_{j}^{n+1}]$, where the values of each in the output array were determined from the exact mathematical solution as follows:

$$z_j^{n+1} = z_j^n \quad \text{for } Cr = 0 \ ;$$

$$z_j^{n+1} = z_{j-1}^n \quad \text{for } Cr = 1.0 \ ; \text{and}$$

$$z_j^{n+1} = z_{j-2}^n \quad \text{for } Cr = 2.0$$

This solution can clearly not be represented by a simple linear expression involving only the weighted sums of the input variables z_{j-2}^{n}, z_{j-1}^{n}, z_{j}^{n} and Cr. Subsequently, an ANN consisting of only linear threshold functions failed to learn how to reproduce these training patterns to anywhere near the degree of accuracy obtained in the previous experiments. Whereas the previous experiments with linear threshold functions and with $Cr = 1$ reduced the residual RMS error during training to less than 10^{-6}, the residual RMS error in this case could not be reduced to much less than 0.04. This corresponded to a coefficient of efficiency, as defined by (4.2.1), of 0.9848. Although these error measures appear to be relatively small in magnitude, their effect upon the performance of the trained ANN may be quite severe.

The effects of these residual errors can be visualised by plotting the results of the verification data sequences. Fig. 5.7 shows the plots of the verification results from the trained ANN for two consecutive time steps when applied to the problem of pure advection of a triangular distribution for three different values of Courant number.

It is obvious from the results that are plotted in Fig. 5.7, that the solution for $Cr = 1.0$ is significantly more accurate than the results for $Cr = 0$ or for $Cr = 2.0$. The three verification tests actually appear to produce almost identical outputs from the ANN in each case. That is, the solution provided by the ANN is insensitive to the actual value of Cr given in the input.

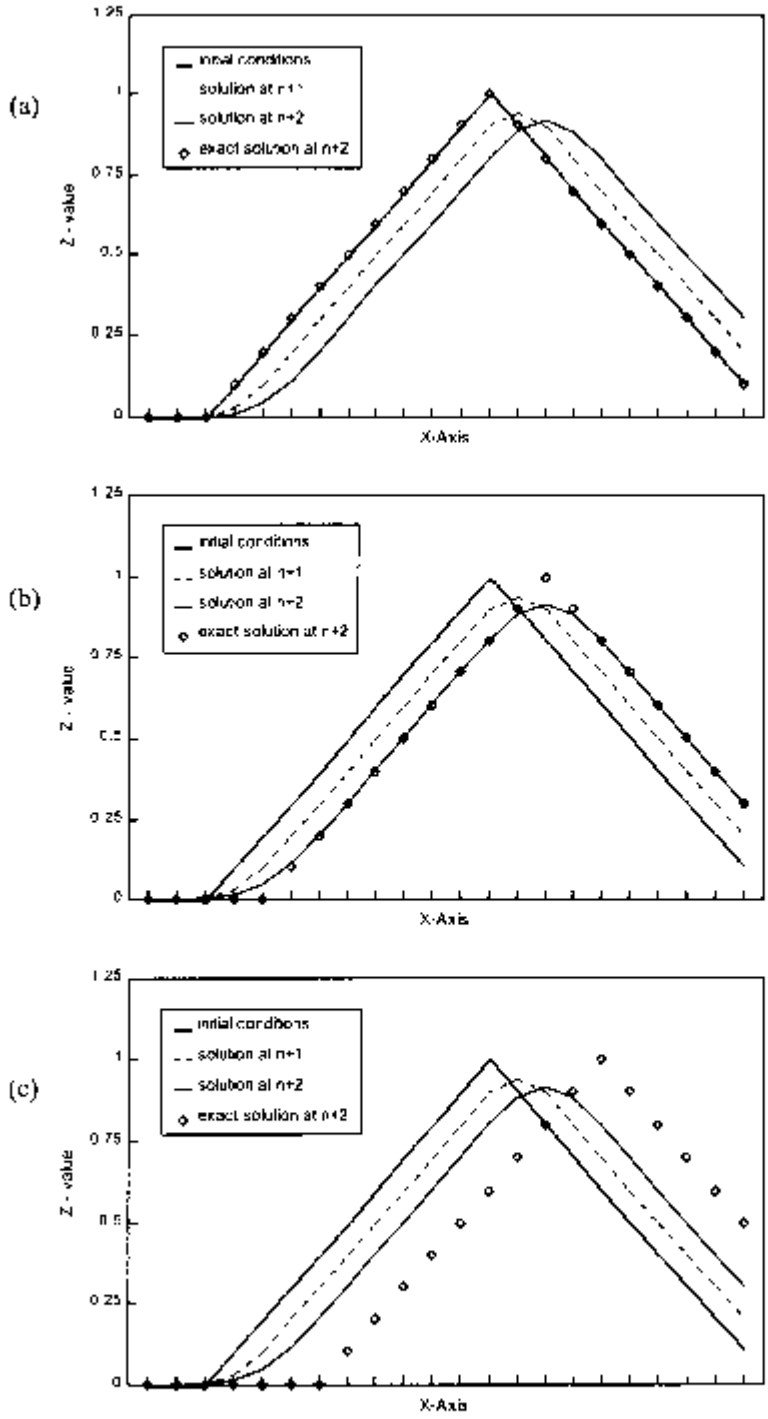


Fig 5.7 Verification results obtained by applying a linear ANN to the problem of pure advection of a triangular distribution for two consecutive time steps, n+1 and n+2, with (a) Cr =0; (b) Cr = 1.0; and (c) Cr = 2.0

To investigate this problem a little further we can consider once more the distribution of weights that were obtained during the training of the ANN, as shown in Fig. 5.8.

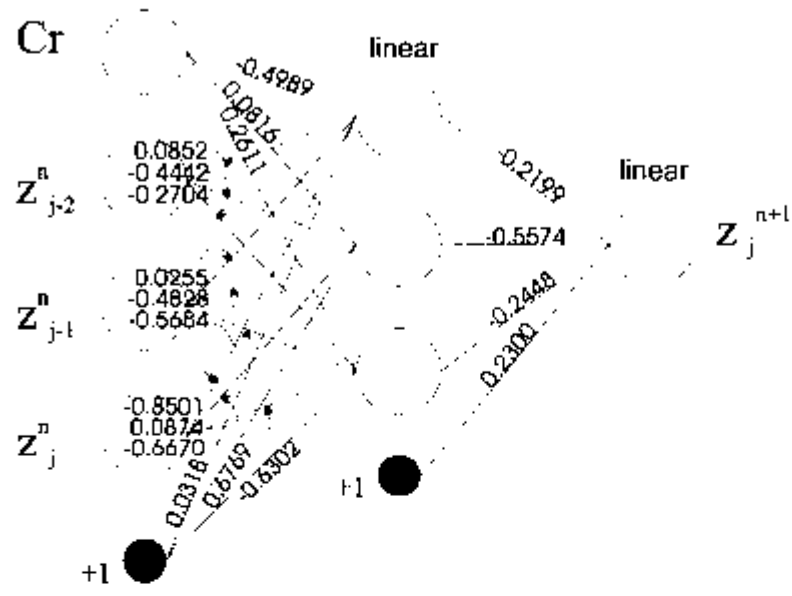


Fig 5.8 Configuration of weights for a 3-layer, linear ANN with four inputs that has been trained to reproduce the pure advection equation for Cr = 0, 1.0 and 2.0

The configuration of the ANN in Fig. 5.8 can be expressed mathematically as:

$$
\begin{aligned}
z_j^{n+1} = -0.2199 \Big[& -0.4989\,Cr + 0.0852\,z_{j-2}^{n} - 0.0255\,z_{j-1}^{n} - 0.8501\,z_j^{n} + 0.0318 \Big] \\
& -0.5574 \Big[0.0816\,Cr - 0.4442\,z_{j-2}^{n} - 0.4828\,z_{j-1}^{n} + 0.0874\,z_j^{n} + 0.6769 \Big] \\
& -0.2448 \Big[0.2611\,Cr - 0.2704\,z_{j-2}^{n} - 0.5684\,z_{j-1}^{n} - 0.6670\,z_j^{n} - 0.6302 \Big] \\
& -0.2300 \Big[+1 \Big]
\end{aligned}
$$

which reduces to:

$$
z_j^{n+1} = 0.0003\,Cr + 0.2951\,z_{j-2}^{n} + 0.4026\,z_{j-1}^{n} + 0.3015\,z_j^{n} + 0.0000 \qquad (5.2.4)
$$

It can be seen quite clearly from the magnitude of the factor associated with the first term on the right-hand side of (5.2.4) why the results shown in Fig. 5.7 were insensitive to the value of Cr. The coefficient associated with the Courant number is only 0.0003, which is very close to zero, so that variations in the value of Cr in (5.2.4) will not lead to any significant variation in the

value of z_j^{n+1}. The entire Cr term has therefore been entirely neglected in the subsequent derivations.

Taylor series expansions of the remaining terms in (5.2.4) about the centre point of the scheme at $(j\Delta x, n\Delta t)$ provide the following expansions:

$$z_j^{n+1} = z_j^n + \Delta t\frac{\partial z}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 z}{\partial t^2} + \text{h.o.t.} \tag{5.2.5}$$

$$z_{j-1}^n = z_j^n - \Delta x\frac{\partial z}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 z}{\partial x^2} + \text{h.o.t.} \tag{5.2.6}$$

$$z_{j-2}^n = z_j^n - (2\Delta x)\frac{\partial z}{\partial x} + \frac{(2\Delta x)^2}{2}\frac{\partial^2 z}{\partial x^2} + \text{h.o.t.} \tag{5.2.7}$$

where 'h.o.t' stands for higher-order terms in the Taylor series expansion.

(5.2.5) - (5.2.7) can be substituted into the remaining terms of (5.2.4) to obtain:

$$\begin{aligned} z_j^n + \Delta t\frac{\partial z}{\partial t} + \frac{\Delta t^2}{2}\frac{\partial^2 z}{\partial t^2} = {} & 0.2951\, z_j^n - 2\Delta x\frac{\partial z}{\partial x} + 2\Delta x^2\frac{\partial^2 z}{\partial x^2} \\ & + 0.4026\left[z_j^n - \Delta x\frac{\partial z}{\partial x} + \frac{\Delta x^2}{2}\frac{\partial^2 z}{\partial x^2}\right] \\ & + 0.3015\, z_j^n + \text{h.o.t.} \end{aligned} \tag{5.2.8}$$

Rearranging the terms in (5.2.8) and dividing by Δt then leads to:

$$\frac{\partial z}{\partial t} + \left(0.9928\frac{\Delta x}{\Delta t}\right)\frac{\partial z}{\partial x} = \frac{-0.0008}{\Delta t}\, z_j^n - \frac{\Delta t}{2}\frac{\partial^2 z}{\partial t^2} + 0.7915\frac{\Delta x^2}{\Delta t}\frac{\partial^2 z}{\partial x^2} + \text{h.o.t.} \tag{5.2.9}$$

Once more, a term has appeared on the right-hand side of this expression, with a coefficient equal to $-0.0008/\Delta t$, which is several orders of magnitude smaller than all other terms in the series. Therefore, we neglect the z_j^n term in (5.2.9) and obtain:

$$\frac{\partial z}{\partial t} + \left(0.9928\frac{\Delta x}{\Delta t}\right)\frac{\partial z}{\partial x} = -\frac{\Delta t}{2}\frac{\partial^2 z}{\partial t^2} + 0.7915\frac{\Delta x^2}{\Delta t}\frac{\partial^2 z}{\partial x^2} + \text{h.o.t.} \tag{5.2.10}$$

If we compare (5.2.10) with the pure advection equation (5.1.1):

$$\frac{\partial z}{\partial t} + c\,\frac{\partial z}{\partial x} = 0 \qquad =(5.1.1)$$

we see that the relationship that has been learned by the ANN is in fact an advection-type equation with a celerity of:

$$c_{ANN} = \left(0.9928\frac{\Delta x}{\Delta t}\right) \qquad (5.2.11)$$

and a truncation error of:

$$\text{T.E.} = -\frac{\Delta t}{2}\frac{\partial^2 z}{\partial t^2} + 0.7915\frac{\Delta x^2}{\Delta t}\frac{\partial^2 z}{\partial x^2} + \text{h.o.t.} \qquad (5.2.12)$$

The truncation error (5.2.12) can be rewritten in terms of a single, second-order space derivative and corresponding numerical diffusion coefficient by substituting the second-order time derivative with the following expression, obtained by differentiating (5.1.1) once with respect to t and once with respect to x and subtracting the two resulting expressions in order to cancel out the cross derivative terms.

$$\frac{\partial^2 z}{\partial t^2} = c^2\frac{\partial^2 z}{\partial x^2} \qquad (5.2.13)$$

Substituting (5.2.13) into (5.2.12) we get:

$$\text{T.E.} = -\left[\frac{-c^2\Delta t}{2} + 0.7915\frac{\Delta x^2}{\Delta t}\right]\frac{\partial^2 z}{\partial x^2} + \text{h.o.t.} \qquad (5.2.14)$$

The celerity of the differential equation represented by the ANN model was given already by (5.2.11) as $c_{ANN} = 0.9928\,\Delta x/\Delta t$ so that we can substitute this expression for c in (5.2.14) to arrive at:

$$\text{T.E.} = \left[0.2987\frac{\Delta x^2}{\Delta t}\right]\frac{\partial^2 z}{\partial x^2} + \text{h.o.t.}$$

which is clearly a diffusion term with a positive diffusion coefficient of $0.2987\ \Delta x^2/\Delta t$. This demonstrates that the most dominant terms in the truncation error will have the effect of introducing diffusion into the solution of the problem. The numerical diffusion coefficient is constant and positive for any choice of (constant) Δx and Δt. This explains why, for the example described in Fig. 5.7, the results appear to be almost identical in each case. Furthermore, a value of $Cr = 1.0$ in this example is equivalent to a physical celerity of $c = 1.0\ \Delta x/\Delta t$. This is practically the same value as the celerity that was given by the ANN model of $c = 0.9928\ \Delta x/\Delta t$, and thus the ANN model gives the most accurate results for $Cr = 1.0$.

The linear ANN used in this example has obviously not been able to generalise the solution of the pure advection problem for the whole range of Courant numbers from 0 to 2.0. In fact, the best performance that the ANN has been able to reproduce is simply the 'average case', represented by $Cr = 1.0$. It was therefore decided to see whether a non-linear ANN would perform any better on this same problem.

Another ANN was then configured using sigmoid threshold functions throughout and trained on the same data as before. The residual RMS error on the training data was reduced significantly in this case from 0.04 to 0.005, giving an improvement in the coefficient of efficiency from 0.9848 in the linear case to 0.9998 in this case. The extreme accuracy of this result can be visualised in Fig. 5.9, which shows a plot of the verification data applied to the trained ANN for two consecutive time steps using $Cr = 1.0$, as well as the solution for one time step using $Cr = 1.5$.

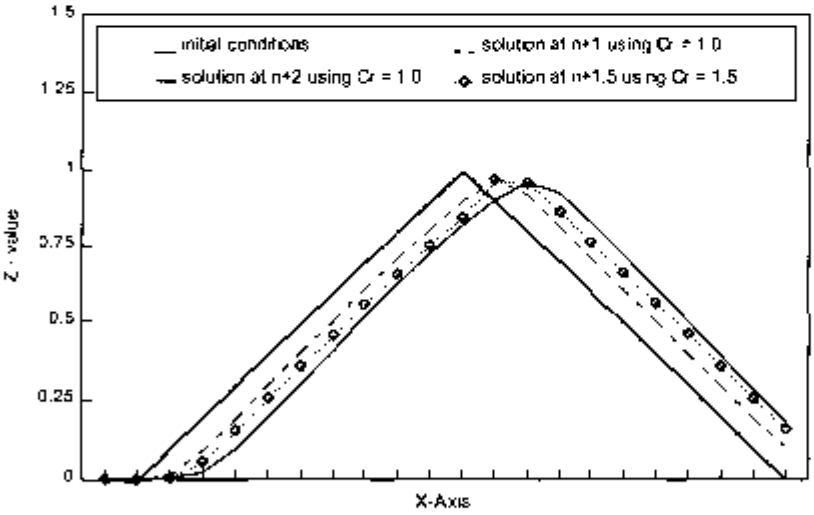


Fig. 5.9 Verification results obtained by applying a non-linear ANN to the problem of pure advection of a triangular distribution with variable Courant number

It is clear from Fig. 5.9 that the solution at time step $n + 1.5$, which represents the result of one calculation with $Cr = 1.5$, lies somewhere between the solutions at time steps $n + 1$ and $n + 2$. The peak value of the triangular distribution should in fact occur halfway between two adjacent gridpoints. Due to the fact that the solution on a discrete grid cannot possibly provide values at

locations between gridpoints, the ANN has had to provide some sort of interpolation between gridpoints where solution values actually do exist. In this case, the ANN is obviously interpolating and generalising quite adequately for Courant numbers between 1.0 and 2.0, which it has not 'seen' before.

To investigate how the error is actually propagating through the solution for both integer and non-integer values of the Courant number, the results of the verification data sequence have also been analysed at an integer step in the time level, for which the exact solution is known. In this way, the effect of the interpolation (that arises when non-integer values of Courant number are used) upon the overall accuracy of the solution can be examined. For this example, the results are compared at time level $n + 3$, which is arrived at after three consecutive calculations using $Cr = 1.0$ and after two consecutive calculations using $Cr = 1.5$. These results of this comparison are depicted in Fig. 5.10.

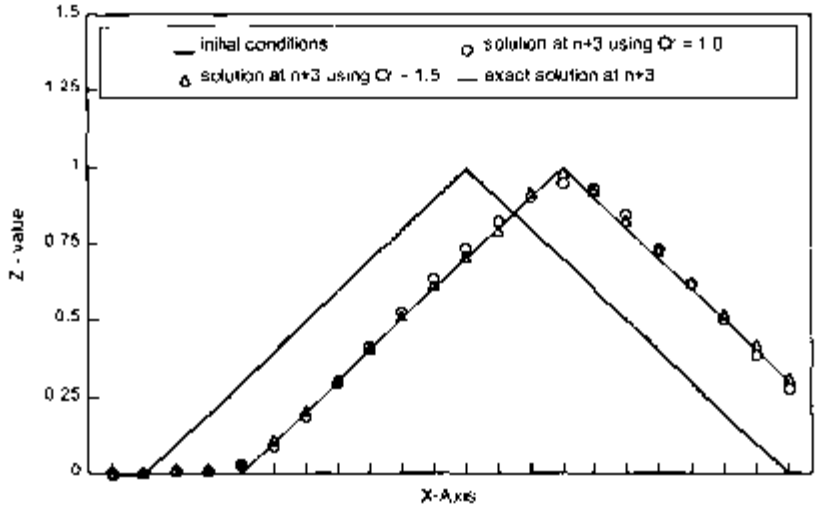


Fig. 5.10 Verification results obtained by applying a non-linear ANN to the problem of pure advection of a triangular distribution at time level n+3 for both Cr = 1 (three calculations) and Cr = 1.5 (two calculations)

Fig. 5.10 also depicts the 'exact' solution of the problem, which is simply the initial conditions translated through a distance of three gridpoints. Both solutions are exceptionally accurate, with a coefficient of efficiency of 0.994 for the solution using $Cr = 1.0$ and a coefficient of efficiency of 0.997 for the solution using $Cr = 1.5$. Once again we also see an amount of 'mass falsification' produced by the ANN model, such that the area under the solution curve has increased from 10.0 to 10.13 for $Cr = 1.0$, an increase of 1.3%, and from 10.0 to 10.28 for $Cr = 1.5$, an increase of 2.8%.

The non-linear ANN has indeed learned a useful relationship for modelling the pure advection process with a variable Courant number to a reasonable degree of accuracy. However, the exact mathematical form of this relationship is practically impossible to determine due to the inherent complexity of the sigmoid function and the number of input nodes and hidden nodes in the network. No such simplified expression can be found that is similar to the one found earlier for

the non-linear ANN in Fig. 5.5. The ANN in Fig. 5.5 had in fact only 9 weighted connections in its entire structure and even included 2 weights in the input layer that were very close to zero and were subsequently neglected. The ANN in the current experiment has a total number of 49 weighted connections, and none of these connections can be neglected. These 49 weights would therefore lead to a much more complex expression than (5.2.3), and the resulting formula would not lend itself to any further physical interpretation.

The problem of a non-constant, local celerity could now be addressed by examining the performance of this ANN model under more 'realistic' flow conditions. If the celerity in a channel should vary in the x-direction, this would mean that, for a given value of Δt and Δx, the Courant number should also vary at each gridpoint. The value of the Cr that should then be used in the input array should be the value of the *locally constant* Courant number. This may even be replaced by the average of the local Cr during the time step n to $n + 1$ and over the gridpoints j, $j - 1$ and $j - 2$.

In order to obtain more realistic flow conditions, flow data was generated using the hydrodynamic modelling system MIKE11 from the Danish Hydraulic Institute. A simple, sloping, rectangular channel, 10 kilometres in length, and with steady, non-uniform flow was instantiated in the hydrodynamic (AD) module of the MIKE11 system. The velocities were calculated at each gridpoint for Δx = 1000 metres. The results of this calculation are given in Table 5.1.

Table 5.1 Velocities, Courant numbers and concentration distribution data for a rectangular channel with steady, non-uniform flow

	gridpoints										
	0	1	2	3	4	5	6	7	8	9	10
velocity (m/s)	1.26	1.32	1.38	1.52	1.64	1.73	2.02	2.21	2.35	2.77	3.31
local Cr	0.76	0.79	0.83	0.91	0.98	1.04	1.21	1.33	1.41	1.66	1.98
average Cr	-	-	0.79	0.84	0.91	0.98	1.08	1.92	1.32	1.47	1.69
initial concentration (mg/l)	0	0	0.25	0.5	0.75	1.0	0.75	0.5	0.25	0	0
MIKE11 concentration after 10 mins	0	-0.03	0.05	0.27	0.50	0.78	0.95	0.83	0.60	0.32	0.15

The velocity distribution given in Table 5.1 was then used in the advection-dispersion (AD) module of the MIKE11 system to calculate the pure advection (i.e. dispersion coefficient equal to zero) of a distribution of concentrations of conservative matter that was being transported by this flow. The initial distribution of concentrations and the solution from the MIKE11 AD module after 10 minutes are also given in Table 5.1. Due to the accuracy and stability requirements of the MIKE11 AD module, the value for Δt that was used in this model could not

exceed 5 minutes in this example. The results in Table 5.1 therefore represent two consecutive calculations of the AD module.

The ANN model in this example required a value of the local Courant number to be calculated for each gridpoint in the input array. Values of Δt = 10 minutes and Δx = 1000 metres were chosen for this experiment, so that the local Cr could be calculated from (5.1.3) at each gridpoint as given in Table 5.1. As mentioned above, a more relevant approximation of the Courant number would be the Cr that was averaged over the gridpoints j, j - 1 and j - 2, which would give a better description of the flow conditions that occur immediately upstream of the calculation point j. This averaged Cr is also given in Table 5.1.

The results of the ANN model after one calculation (i.e. after one time step of 10 minutes) are plotted in Fig. 5.11 for both the local Courant number and the locally-averaged Courant number. The initial concentration distribution and the results from the MIKE11 AD module are also depicted in this figure.

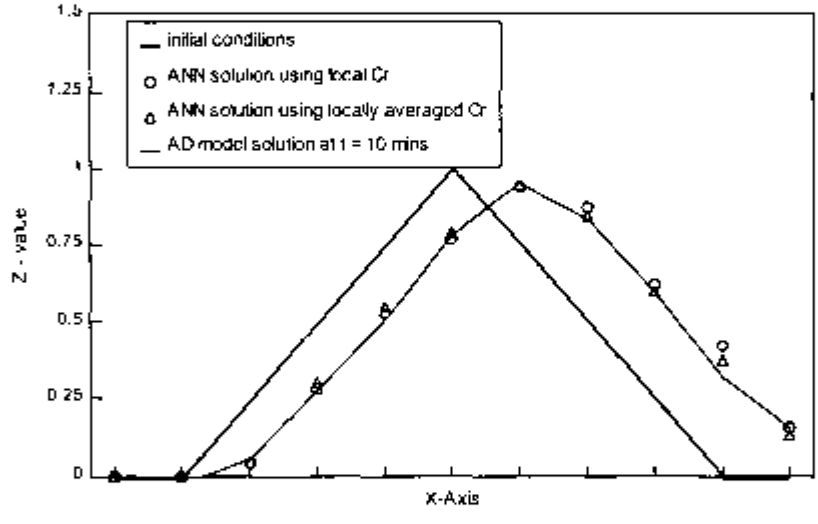


Fig. 5.11 Verification results of an ANN model for pure advection of a triangular concentration distribution under steady, non-uniform flow conditions

The use of the locally-averaged Courant number appears to give a slight improvement in the results, as expected. In this example, the root-mean-square (RMS) error between the MIKE11 results and the two ANN models were 0.036 and 0.027 for the local Cr and for the locally-averaged Cr models respectively. The ANN model has therefore provided an exceptionally accurate solution of the pure advection scheme using a time step that was twice as large as the MIKE11 model. Although there may be some degree of mass falsification in the ANN model, it is only relatively small in this case. Furthermore, the results of the ANN model contain only positive values for concentration, whereas the MIKE11 models gives small negative values at the extremes of the concentration distribution, as seen in Table 5.1.

6 Data Mining

6.1 General

Data mining constitutes one part of the multi-step knowledge discovery process for extracting useful patterns and models from raw data stores. Fayyad *et al* (1996, p. 44) describe a variety of data mining procedures that include:

- *classification*: in which a function is learned that maps (classifies) a data item into one of several predefined classes.
- *regression*: in which a function is learned that maps a data item to a real-valued prediction variable.
- *clustering*: in which one seeks to identify a finite set of categories or clusters to describe the data.
- *summarisation*: in which a compact description is found for a subset of the data.
- dependency modelling: in which a model is found that describes significant dependencies between variables.
- *change and deviation detection*: which focuses on discovering the most significant changes in the data from previously measured or normative values.

The other steps in the knowledge discovery process are those of data preparation, data selection, data cleaning, the incorporation of appropriate prior knowledge and proper interpretation of the discovered results. As was mentioned in §1.3, cultural knowledge is only useful if it is in a form that can be accessed and used reliably by different people. The raw data involved here are usually so numerous that the more traditional, manual methods of data mining must now make way for computer-based methods, which are much better suited to unearthing meaningful patterns and structures from vast databases rapidly and reliably.

The goal of the knowledge discovery steps may be simply to condense the data into a short, printed report, or it may be to find a model of the process that generated the data and which may be used to estimate values in future cases. The fitted models play the role of inferred knowledge. This is similar to the problem of systems investigation, or systems identification, defined earlier in §4.1 as the field of study which is concerned with the direct solution of technological problems subject only to the constraints imposed by the available data and so not subject to 'physical' considerations (Amorocho and Hart, 1964).

Following Iba *et al.* (1993), we may define systems identification in the following way. Imagine that a system produces an output value, y, and that this y is dependent on m input values, thus:

$$y = f(x_1, x_2, x_3, \ldots, x_m) \tag{6.1.1}$$

Given a set of N observations of input-output tuples, as shown Table 6.1, the system identification task is to approximate the true function f with $\bar{f}$.

Table 6.1 A input-output example set

INPUT				OUTPUT
x_{11}	x_{12}	...	x_{1m}	y_1
x_{21}	x_{22}	...	x_{2m}	y_2
...	...	...	...	...
x_{N1}	x_{N2}	...	x_{Nm}	y_N

Once this approximate function $\bar{f}$ has been estimated, a predicted output y can be found for any input vector $(x_1, x_2, ..., x_m)$ from:

$$\bar{y} = \bar{f}(x_1, x_2, x_3, ..., x_m) \tag{6.1.2}$$

The $\bar{f}$ is called the *complete form* of f. As was mentioned in Chapter 1, and as demonstrated repeatedly in Chapters 4 and 5, artificial neural networks constitute one class of sub-symbolic paradigm that lends itself very easily to the problem of searching for and storing the complete form of f.

An ANN is an electronic knowledge encapsulator which encapsulates its knowledge at the level of the *taxinomia* by establishing some useful relationship between a collection of signs on the input side and a collection of signs on the output side. The actual relationship is stored electronically at the sub-symbolic level as a series of weights and connections between nodes. It is usually not possible to extract and interpret an exact, symbolic, mathematical formulation of this relationship. At the level of the *mathesis*, this relationship becomes extremely complex due to the non-linear nature of the transformations that take place upon the weighted sums of the signals through the application of sigmoid threshold functions.

In §5.2 it was shown that, under very special circumstances that allow the use of linear threshold functions, it was possible to derive a physically-realistic, mathematical representation of the encapsulated relationship in the form of the differential equation (5.2.10). Unfortunately, ANNs with linear threshold functions do not find much use in practice, so that this type of analysis cannot generally be applied.

Artificial neural networks may therefore provide an extremely powerful paradigm for only some of the data mining procedures mentioned above (e.g. classification and regression). In some other procedures, the use of the existing types of ANNs may be entirely inappropriate. This chapter describes the performance of ANNs when applied to various problems of data mining and system identification. These results are compared to the results obtained from more traditional, manual

methods, as well as some other, computer-based methods. In particular, a comparison is made to results obtained from another sub-symbolic approach called genetic programming (GP).

As described in §3.2, GP is one of the evolutionary algorithms that can actually generate models in a symbolic form. Whereas traditional genetic algorithms typically operate by combining binary strings which encode real-valued independent variables (*see*, for example, Babović (1993)) in the case of GP, the symbolic expressions themselves are subject to the genetic operators of recombination and mutation (*see* Babović and Abbott, 1997). In this way, GP may, at first, appear to be entirely symbolic in nature. This is, however, not at all true. As explained by Babović (1996, p. 248), both ANNs and GPs are sub-symbolic in the sense that a manipulation of data occurs at a level which is below that of the symbol. The tokens which are manipulated are, at best, *indicative signs*, but do not have any *expressive* capability in and by themselves. GP manipulates tree structures that only acquire any meaning, or semantic content, once the tree has been interpreted as an algebraic expression in Reverse Polish Notation, or prefix notation, of standard computer science (*see* Babović and Abbott, 1997, p. 402).

6.2 Rainfall-runoff Modelling

Babović (1996, *but see also* Babović and Abbott, 1997) performed data mining experiments using genetic programming on the same rainfall-runoff data that was generated for the artificial catchment experiments described in §4.4. Even though experiments were performed over the entire range of artificial catchments, from the extremely linear to the extremely non-linear case, the following section only describes the results from the moderately non-linear catchment, referred to in the earlier section as the regular catchment. The reason for this is that the results obtained by Babović for the regular catchment are considered to be quite representative of the typical performance of GP when applied to this type of hydrological data analysis.

The training data for the GP consisted of the concurrent and fourteen antecedent rainfall depths (i.e. $[\, r_t, r_{t-1}, r_{t-2}, \ldots, r_{t-14} \,]$), as well as two antecedent flow values (i.e. $[\, q_{t-1}, q_{t-2} \,]$). The output data consisted only of the concurrent flow value q_t. This is very similar to the training data configuration that was used for the ANN model, which used the concurrent and fourteen antecedent rainfall depths and three antecedent flow ordinates..

After a sufficient number of generations of the GP, two expressions were generated that fitted the training data with rather similar accuracy. Babović (1996) gives these two expressions as:

$$q_t = 0.1052\left[q_{t-1} - 1.2993r_{t-1} + 0.032\right] + q_{t-1} - 0.6636r_{t-11} - 0.0031 \qquad (6.2.1)$$

and

$$q_t = 0.10\left[q_{t-1} - 0.201r_{t-10} - 0.0032\right] + q_{t-1} - 0.7214r_{t-11} - 0.0032 \qquad (6.2.2)$$

These two equations can be simplified to give:

$$q_t = 1.1052q_{t-1} - 0.1367r_{t-12} - 0.6636r_{t-11} + 0.0003 \qquad (6.2.3)$$

and

$$q_t = 1.1000q_{t-1} - 0.7214r_{t-11} - 0.0201r_{t-30} - 0.0035 \qquad (6.2.4)$$

That is, the GP has found that the entire rainfall-runoff process can be described by a superposition of pure advection processes. (For a more detailed discussion of this interpretation see Babović and Abbott ,1997, p. 416). Purely from the viewpoint of data mining, the GP has found that only three input variables are necessary to describe the rainfall-runoff modelling process in each case. Regardless of whether one accepts expressions (6.2.3) and (6.2.4) as being 'physically realistic' or not, the GP has achieved one of the primary goals of data mining - that of finding a compact description of the data set. It is now interesting to compare the performance of expressions (6.2.3) and (6.2.4) with the solution obtained by the 3-layer ANN that was trained on the same data, as described in §4.4. The coefficients of efficiency for the training data sequence and for the so-called 'normal' verification data sequence are given in Table 6.2 for the ANN model and for both of the GP expressions.

Table 6.2 Performance overview in terms of coefficients of efficiency for an ANN model and for two GP expressions on training and verification data from a regular catchment

model	coefficients of efficiency	
	training data sequence	'normal' verification data
3-layer ANN	0.9987	0.9941
GP expression (6.2.3)	0.9903	0.9926
GP expression (6.2.4)	0.9900	0.9924

Table 6.2 indicates that efficiencies of more than 99% have been achieved for both methods in fitting a function to the given data. The ANN model performs slightly better than the GP expressions on both training and verification data; however, the ANN model requires 18 input variables, whereas the GP expressions require only 3 input variables each.

The interpretation of the ANN model was given earlier when it was concluded that the total time interval of the window of input rainfall data should encompass the centroid to centroid lag times of the catchment runoff data. The GP expressions then confirm this conclusion by selecting only the rainfall ordinates with similar lag times. All other rainfall ordinates have been neglected.

It is not possible here, nor even very prudent, to draw conclusions about the superiority of the one method over the other. The advantage of the extra accuracy of the ANN approach in one

application may, for example, be counteracted by the advantage of the extreme compactness of the GP expressions in another. In fact, a truly superior method is most likely to be obtained when the two techniques are combined in a so-called 'hybrid' approach. This approach can be demonstrated by considering in more detail how the strengths of each separate technique can be used to enhance the performance of the other.

In the above example, the use of 15 rainfall ordinates in the input data set to the GP was decided upon after several ANN models with different configurations had been instantiated and tested to discover the optimal length of input window. This initial testing of a number of variations can be done much more rapidly with an ANN than with GP, due to the extreme simulation time required for every application of the GP algorithm as compared to the training time of an ANN. Having established this optimum window length, the input data to the GP was then selected and the resulting expressions (6.2.3) and (6.2.4) resulted from the evolution process.

However, (6.2.3) and (6.2.4) can now be used to further enhance the performance of the ANN. For example, expression (6.2.3) indicates that a very accurate solution can be found with only the three input variables; r_{t-11}, r_{t-12}, and q_{t-1}. This information was therefore used to configure a new ANN model with only these three input variables. The results of using an ANN model with 5 hidden nodes, and using sigmoid threshold functions throughout, produced coefficients of efficiency of 0.9915 for the training data and 0.9933 for the validation data, which can be compared to the coefficients of efficiency given in Table 6.2. It is seen that these new results are only slightly less accurate than the original ANN model, however, they are still more accurate than the GP expressions, but now utilising the same amount of input data as the GP model. The new ANN model is of course much more compact than the original model and is subsequently much easier and faster to train.

Lastly, it is also possible to use the power of the ANN learning algorithm to improve directly upon the given GP expressions. The linear nature of the GP expressions means that these expressions can be exactly represented by a simple two-layer ANN which uses linear threshold functions. Two linear ANNs were therefore configured and trained using $[r_{t-11}, r_{t-12}, q_{t-1}]$ as input data in the first case, and $[r_{t-10}, r_{t-11}, q_{t-1}]$ as input data in the second case. These small, linear ANNs converged very rapidly using the back-propagation learning algorithm to provide the solutions:

$$q_t = 1.0962q_{t-1} - 0.5238r_{t-12} - 0.1812r_{t-11} - 0.0845 \tag{6.2.5}$$

in the first case, and:

$$q_t = 1.0970q_{t-1} - 0.6478r_{t-11} - 0.0320r_{t-10} - 0.1090 \tag{6.2.6}$$

in the second case.

The coefficients of efficiency for expressions (6.2.5) and (6.2.6) are summarised in Table 6.3, which can be compared to the results in Table 6.2 for expressions (6.2.3) and (6.2.4).

Table 6.3 Coefficients of efficiency for linear ANN models on training and verification data from a regular catchment

model	coefficients of efficiency	
	training data sequence	'normal' verification data
ANN expression (6.2.5)	0.9911	0.9929
ANN expression (6.2.6)	0.9902	0.9923

The most significant result of this experiment is the extreme similarity that exists between the GP expression (6.2.4) and the ANN-induced expression (6.2.6). The coefficients of efficiency for these expressions differ only very slightly and thus indicate that both the GP and ANN have found the optimum values of the coefficients in the linear expression that relates these three input variables. The more significant differences between the GP expression (6.2.3) and the ANN-induced expression (6.2.6) indicate that the GP had not yet arrived at the optimum values for the coefficients in this case. This is not necessarily due to an error in the GP algorithm, but rather indicates that the evolutionary process for this particular expression was halted prematurely.

6.3 Salt Intrusion in Estuaries

The results in this section are taken from the previously published paper of Babović and Minns (1994).

The management of estuarine water resources requires as a central instrument accurate models of the salt intrusion length and the longitudinal distribution of the salinity as a function of some directly measurable parameters such as the geometry of the estuary, fresh-water flow and tide. There have been quite a number of models developed to serve this purpose, but almost all of these models have some drawbacks. The existing models are mainly based on laboratory experiments and are either inaccurate or demand rather specific data that are both difficult and expensive to collect.

A novel technique for the rapid assessment of salt intrusion in alluvial estuaries has been presented by Savenije (1992). This work describes a methodology which is readily implemented and which allows engineers to determine the degree of salt intrusion on the basis of a minimum amount of available information. The method is based on the full equations for motion and conservation of mass of water, salt and sediment in which a number of assumptions are made with respect to estuary geometry, tidal hydraulics and the mixing mechanism. The method is predictive in the sense that it allows the user to predict the salt intrusion as a function of fresh water inflow into the estuary and vice versa. For this purpose, empirical relations were derived that relate model parameters to measurable estuary parameters.

The rapid assessment technique can be applied to both steady and unsteady state situations and to estuaries where evaporation plays an important role. The technique for steady state has been

used and applied successfully to 52 surveys in 16 different estuaries world wide. The data and the performance of various salt intrusion formulae are summarised in Savenije (1993a,b).

The most important output of this predictive model is the salt intrusion length L, this being the distance from the estuary mouth to the point where the salinity level reaches the river salinity level. Obviously, the intrusion length will vary during a tidal cycle, and so, subsequently, the most descriptive variable in this case has been taken to be the salt intrusion length at high-water slack, L^{HWS}

Savenije (1993a,b) gives an expression for L^{HWS} as:

$$L^{HWS} = a \ln\left(\frac{1}{\beta}+1\right) \tag{6.3.1}$$

where:

$$\beta = \frac{Ka}{\alpha_0 A_0} \tag{6.3.2}$$

with

$$\alpha_0 = -\frac{D_0}{Q_f} \tag{6.3.3}$$

and:

A_0 = cross-section area at the estuary mouth;
a = cross-section area convergence length;
D_0 = dispersion at the estuary mouth;
K = Van der Burgh's coefficient (defined by Savenije, 1993a,b);
Q_f = fresh water discharge.

In some estuaries the geometry of the estuary can only be described sufficiently by using two exponential functions, one being valid up to the inflection point, and the other beyond this point. In this case the expression for salt intrusion length becomes:

$$L^{HWS} = x_1 + a_2 \ln\left(\frac{1}{\beta_2}+1\right) \tag{6.3.4}$$

where:

$$\beta_2 = \frac{Ka_2}{\alpha_1 A_1} \tag{6.3.5}$$

with

$$\frac{\alpha_1}{\alpha_0} = 1 - \beta_1\left(\exp\left(\frac{x_1}{a_1}\right) - 1\right) \tag{6.3.6}$$

and:

a_1 = cross-section convergence length for main branch;
a_2 = cross-section convergence length for secondary branch;
x_1 = distance from the mouth to the inflection point.

As can be seen from (6.3.1) and (6.3.4), the application of this methodology to any general estuary would require considerable *a priori* knowledge of the physical characteristics of the estuary, such as, for example, the number of branches that need to be defined in order to describe the estuary shape adequately.

Ideally, a method of this sort should be general enough to be used under any circumstances. The ideal solution should therefore consist of a single formulation, containing variables which are universal to all estuaries, and not requiring additional expressions depending on anomalous geometries. In pursuit of this ideal, Babović and Minns (1994) attempted to duplicate the above results by applying ANNs and GPs in order to arrive at simpler, more general and, hopefully, more accurate expressions.

A three-layer ANN was constructed and trained to relate an input pattern consisting of all of the salt intrusion parameters, A_0, K, a, Q_f and α_0, to an output pattern consisting of only the salt intrusion length - L^{HWS}. The input layer of the network consisted of five nodes (one for each parameter) and the hidden layer consisted of five nodes. The ANN converged to its most accurate solution after some 12000 iterations of the input data.

The results of the applying the ANN to the data given by Savenije (1993) are depicted in Fig. 6.1. This figure shows the actual measured salt intrusion length for 45 steady-state measurements in 15 estuaries worldwide. These measured data are also compared to the results obtained by Savenije using expressions (6.3.1) and (6.3.4).

The GP resulted in the following expression for L^{HWS}:

$$L^{HWS} = \frac{2A_0}{Q_f} - Q_f - A_0\left[\ln(K) + 1\right] + \ln(a) + 102.26\times10^4 \tag{6.3.7}$$

The results of using expression (6.3.7) to predict L^{HWS} are also illustrated in Fig. 6.1. It should be noted that the evolution towards the solution (6.3.7) was halted in this case after some 12000 generations due to time limitations. Therefore (6.3.7) cannot be said to represent the end result of the process of evolution, and hence the final solution. Indeed, as in any non-trivial evolutionary process, it is impossible to identify the end of this process. Evolution, whether natural or computational, is not a purposive or directed process and there is certainly no scientific evidence, for example, to support the assertion that the only goal of natural evolution is to produce *Homo Sapiens*!

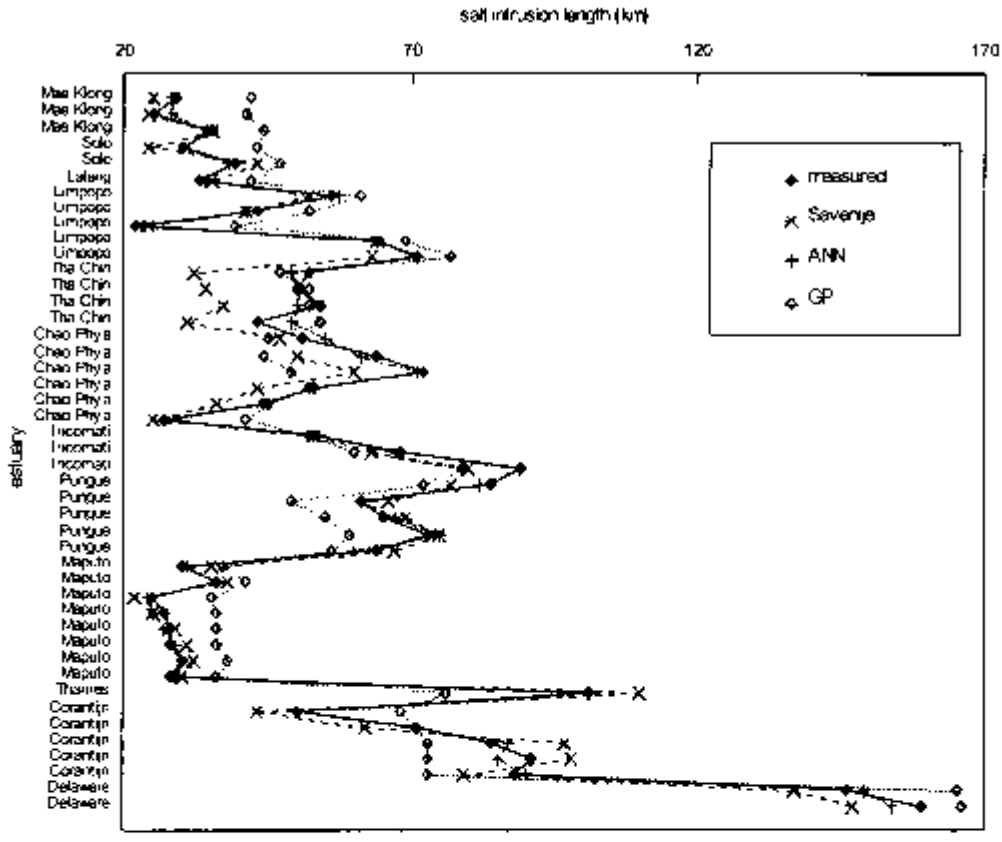


Fig. 6.1 Results of Savenije (1993a,b), an ANN and GP compared to 45 measurements of salt intrusion length from 15 different estuaries worldwide

An error measure was introduced in order to quantify the accuracy of each of the modelling techniques. One measure of the accuracy is the absolute error between the measured salt intrusion length and the predicted salt intrusion length. A measure of error, ε, has therefore been adopted that is calculated as the average absolute error between the predicted and measured salt intrusion length, averaged over the 45 sets of data. The overall, average errors, ε, for each method are given in Table 6.4.

Table 6.4 Average absolute errors for each method in predicting the salt intrusion length for 15 different estuaries worldwide

Method	Absolute error, ε (km)
Savenije	6.2
ANN	1.6
GP	9.9

A more satisfactory method for visualising the overall performance of each of the methods in their representation of the measured data is to use a scatter plot. The scatter plot for this data is shown in Fig. 6.2.

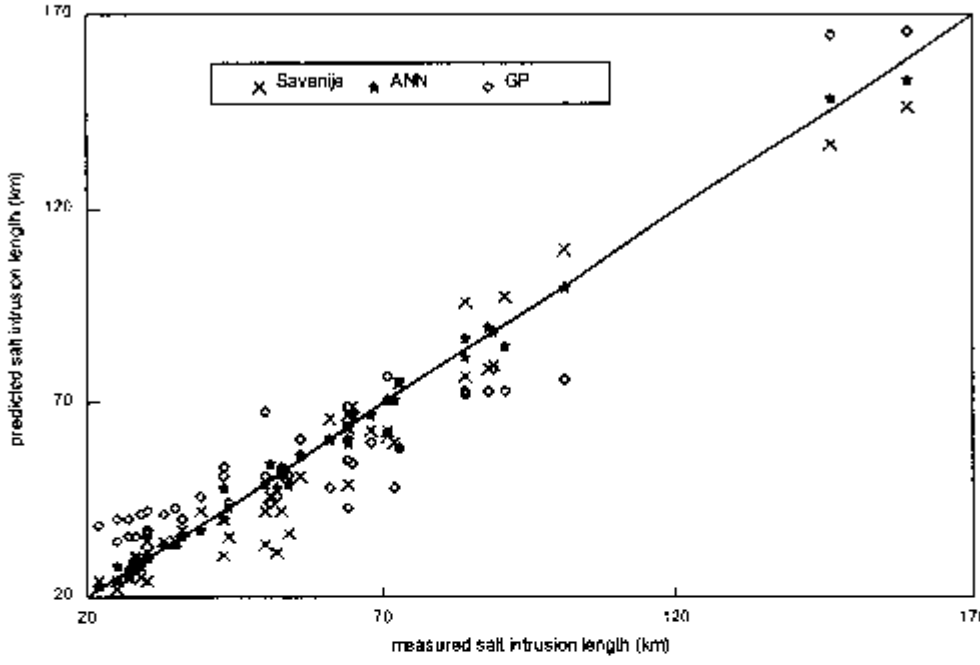


Fig. 6.2 Scatter plot showing the deviation of the results of the various methods from the actual measured values

Figs. 6.1 and 6.2 and Table 6.4 indicate that ANN and GP methodologies are certainly just as accurate as traditional, empirically-based methods for discovering usable salt-intrusion-length relationships based only upon measured data in estuaries. In the case of the ANN we see that the error is extraordinarily small compared to all of the other methods. On the other hand, the ANN does not produce a symbolic-algebraic expression of the complete form of the solution. As such it does not allow the modeller to gain insight directly into the physical problem. It does, however, confirm the existence of a strongly deterministic relationship between the five measured input quantities and the salt intrusion length at high-water-slack.

Contrarily, GP produces an expression that is slightly less accurate than the empirically-based results but is, nevertheless, a symbolic-algebraic expression that highlights the importance of only the most significant variables in the calculation of the salt intrusion length. This result demonstrates the power of GP to produce new expressions that may never have been considered previously by the modeller.

The results of both Savenije and GP demonstrate quite clearly that, if sufficient measurements are made of a sufficient number of relevant parameters, it is possible to derive quite complex, empirical and physically-based formulae that, through the enormous calculating capacity of modern computers, can be solved very rapidly and accurately. However, the problem with this

type of approach, is the extreme complexity of the combinations of parameters that appear in the derived relationships which, even if physically-based, do not really provide a great deal of engineering insight into the problem at hand due to the fact that these expressions are very difficult to interpret in terms of a description in a natural language.

Finally, one of the most significant strengths of the results provided by the sub-symbolic paradigms is that the trained ANN and the GP expression (6.3.7) are valid for all 45 sets of data, which means that the results obtained can be applied directly to all of the above estuaries. In contrast, the empirically-based results of Savenije involve the occasional use of either expression (6.3.1) or (6.3.4), depending on the estuary under consideration, which requires an extra measure of knowledge and insight on the side of the modeller in order to select the most appropriate relationship in each case.

6.4 Sediment Transportation

The results of this section are taken from the previously published paper of Minns (1995). The data sets used in this study were taken from the work of Zyserman and Fredsøe (1994), which were in turn derived from Guy *et al.* (1966). This work involved the determination of the bed concentration of suspended sediment c_b from flume experiments. The experimental data provided the total, steady-state sediment load for a range of hydraulic conditions including varying discharge, bed-slope and water depths. The measurement of the concentration of suspended sediment is extremely difficult near to the bed due to the large vertical gradients in the concentrations that occur very close to the bed. A further problem is that of trying to separate the bed load from the suspended load.

Zyserman and Fredsøe calculated the suspended sediment load q_s by subtracting the bed-load, calculated from the Engelund-Fredsøe (1976) formulation, from the total load. c_b could then be determined from q_s by applying Einstein's (1950) formulation in which the suspended sediment concentration profile is integrated over the depth down to the lower limit of the suspended sediment layer located at a distance of twice the grain diameter from the bed.

In order to derive an expression relating the extremely-difficult-to-measure bed concentration, c_b, to other, easy-to-measure, hydraulic parameters, Zyserman and Fredsøe applied dimensional analysis techniques to all of the available measured data described above. The hydraulic conditions of each experiment could then be represented by the derived Shields parameters ,θ and θ', given by:

$$\theta = \frac{u_f}{(s-1)gd} \tag{6.4.1}$$

and

$$\theta' = \frac{u_f'}{(s-1)gd} \tag{6.4.2}$$

where:

u_f – shear velocity = $\sqrt{g D I}$;
s – relative density of sediment;
d – d_{50} – median grain diameter;
D = average water depth;
I = water surface slope;
u_f' = shear velocity related to skin friction = $\sqrt{g D' I}$
D' = boundary layer thickness defined by:

$$\frac{v}{u_f'} = 6 + 2.5 \ln\left(\frac{D'}{k_N}\right) \tag{6.4.3}$$

v = mean flow velocity;
k_N = bed roughness = $2.5d$

Various combinations of the Shields parameter were considered together with various forms of the Rouse number, z, given by:

$$z = \frac{w_s}{\kappa u_f} \tag{6.4.4}$$

where:

w_s = settling velocity of suspended sediment;
κ = von Karman's constant (= 0.4)

The resulting relationship arrived at by Zyserman and Fredsøe (1994) was given as:

$$c_b = \frac{0.331\ (\theta' - 0.045)^{1.75}}{1 + \frac{0.331}{0.46}\ (\theta' - 0.045)^{1.75}} \tag{6.4.5}$$

Equation (6.4.5) implies that the *only* significant parameter for determining the bed concentration is θ'. The discussion of Eq. (6.4.5) given by Zyserman and Fredsøe indicates that this expression gives comparable, if not better, accuracy than several other, more complex, empirical formulations. The question remains, however, whether this result could be improved by including *all* of the measured data instead of only one parameter. Artficial neural networks can be used very easily to assess this hypothesis.

A set of parameters that actually incorporates all of the measured data includes θ, θ', u_f, u_f', d_{50}, and w_s. An ANN was therefore set up with six nodes in the input layer, one node for each of the parameters given above, four nodes in the hidden layer and a single node in the output layer consisting of the bed concentration, c_b. The ANN was trained using all of the data that was available to Zyserman and Fredsøe. The results of the training are depicted in Fig. 6.3, which is a scatter plot of the measured bed concentrations compared to the bed concentrations as calculated by the trained ANN and by Eq. (6.4.5).

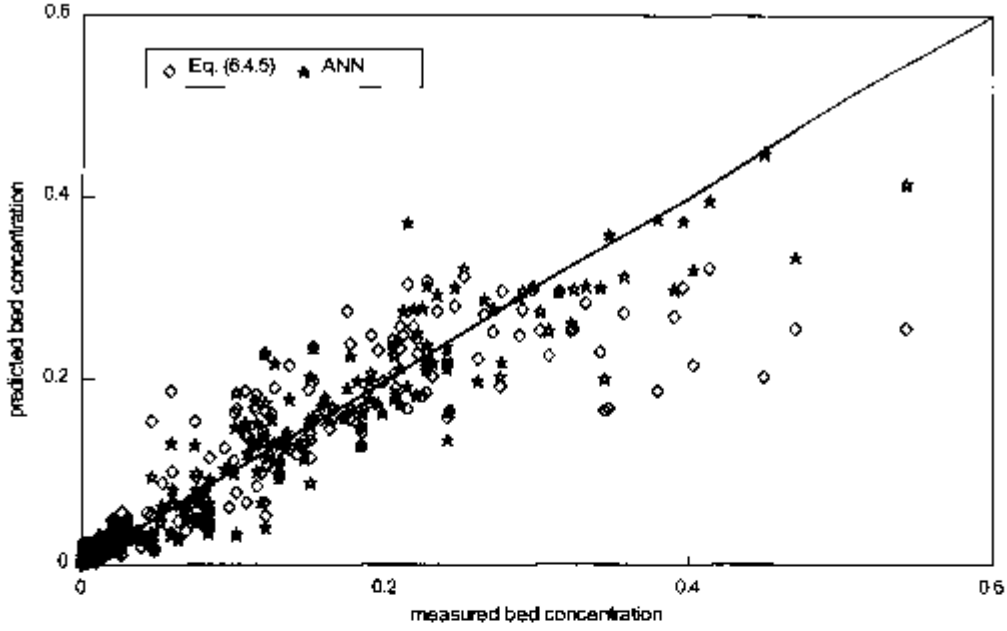


Fig 6.3 Scatter plot of bed concentrations calculated by an ANN with six input parameters compared to results from Eq. (6.4.5)

It can be seen from Fig. 6.3 that the ANN has a smaller scatter from the diagonal compared to the results from Eq. (6.4.5), especially for the higher values of bed concentration. To compare the accuracy of the ANN to that of Eq. (6.4.5), the root-mean-square (RMS) error was calculated for each method. The RMS error for the ANN was 0.034 and the RMS error for Eq. (6.4.5) was 0.049, which was equivalent to an increase in the coefficient of efficiency from 0.802 to 0.905. This indicates a significant improvement in the predicting capability on the part of the ANN, as compared to the more traditional approach. Furthermore, the ANN makes use of all of the available data and hence provides a solution that will be sensitive to variations in any of the hydraulic parameters and not just θ' as is the case with Eq. (6.4.5).

An ANN has therefore demonstrated the capacity to discover and learn a relationship between easy-to-measure hydraulic flow parameters and the bed concentration of suspended sediment with significantly more accuracy than that achieved by more traditional regression analysis and dimensional analysis techniques. Furthermore, whereas dimensional analysis may reject some hydraulic parameters in order to simplify the resulting expressions, the ANN makes use of all of the available measured data, thus improving the accuracy and sensitivity of the resulting relationship without the need for any preliminary analysis to select the most significant parameters, or to disregard less significant parameters.

Interestingly enough, if the ANN used above is simplified and trained to calculate c_b from only one input parameter, namely θ', then results are obtained that are of very similar accuracy to Eq. (6.4.5). For the sake of comparison, an ANN was trained with only this one input parameter. The scatter plot of the results is illustrated in Fig. 6.4. In this case, the ANN results have an RMS

error of 0.048 and the scatter of points about the diagonal line in Fig. 6.4 is almost exactly comparable to the results from Eq. (6.4.5).

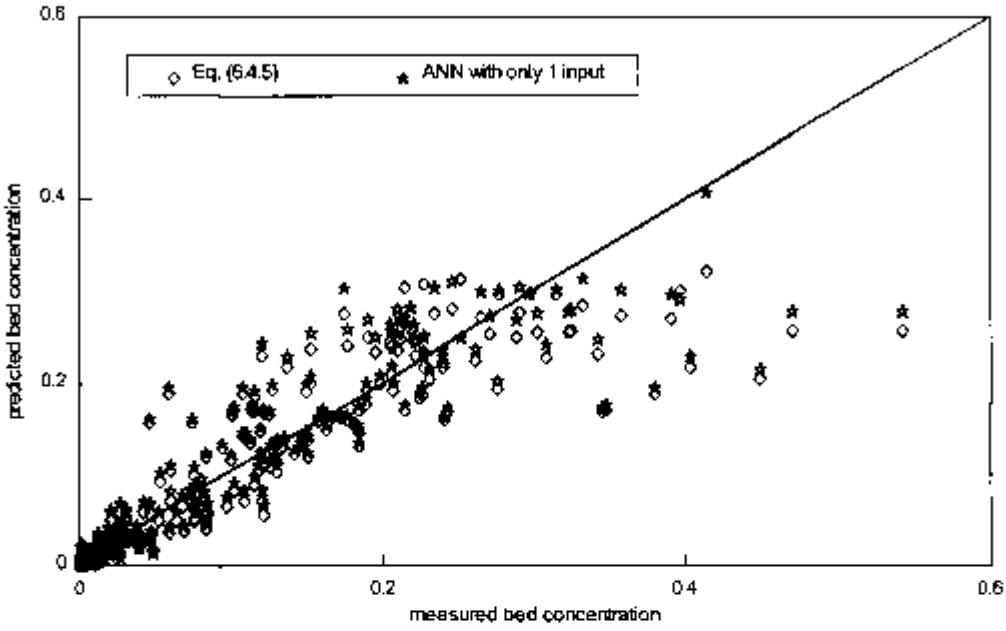


Fig. 6.4 Scatter plot of bed concentrations calculated by an ANN with only one input parameter compared to results from Eq. (6.4.5)

Now, if Eq. (6.4.5) is accepted as having no deeper physical meaning (i.e. it has no semantic content), so that it is only a computational tool to calculate c_b, then the only major difference between this computational tool and the ANN is that Eq. (6.4.5) can be written down exactly on paper while the ANN is stored, usually electronically, as a series of weights and connections between nodes. The evaluation of Eq. (6.4.5), however, also requires the use of some sort of modern computational device and so the restriction of ANNs to be used only on computers is not considered as an extraordinarily limiting factor here.

Lastly, a GP analysis of this same data is also described in Babović and Abbott (1997). In this case, the GP performs slightly better than the dimensional analysis approach, but produces algebraic expressions of remarkably similar form to Eq. (6.4.5) that again utilise only some of the six parameters that encapsulate all of the measured data. The best performing GP expression can be written in algebraic form as:

$$c_b = \frac{0.331\left(\theta' - \dfrac{\theta}{w_s}\right)^{1.75}}{0.65\,(\theta' - 0.403\sqrt{d_{50}})^{1.60}} + 1.11 \qquad (6.4.6)$$

Eq. (6.4.6) actually only improves the coefficient of efficiency from 0.802 to 0.816. Once again, the GP has found the simplest expression that contains only the 'most significant' parameters in the relationship. If we compare the accuracy of this solution to the accuracy of the ANN model achieved above (i.e. coefficient of efficiency of 0.905), it becomes clear that the exclusion of even only some of the 'less significant' parameters can still affect the overall accuracy of the final solution quite significantly when dealing with the more complex physical relationships, typical of those that exist in sediment transportation problems. In effect, the GP approach can become subjected to the same limitations of a restricted symbol system as does any algebraic system, whereas the ANN, being much 'more sub-symbolic', largely escapes from this restriction. For a more complete discussion of the problems of reading a physical 'meaning' into Eq. (6.4.6), or, for that matter, even Eq. (6.4.5), reference is made to the paper of Babović and Abbott (1997, pp. 416-420).

7 Conclusions

This present work, like all research works, is really only a kind of progress report. Research itself is like a rugged landscape, punctuated by numerous local optima. Exploring our way through it, we may choose to stop, temporarily trapped at these local optima, to rest and survey the hills around, and to write a publication or two. A movement in the direction of a new hill delineates the definition of a new research programme, an elaboration of fragments of existing works, and an exchange of ideas. This present work, then, is an amalgam of various efforts directed towards a definition of a unified view on sub-symbolic paradigms and, in particular, the suitability of one specific such paradigm, that of artificial neural networks (ANNs) for use within a hydroinformatics framework.

In Chapter 4, the ability of an ANN to relate a runoff ordinate to the pattern of antecedent rainfall depths was demonstrated. In hydrological modelling terms, the ANN does not identify a form of model, such as the non-linear reservoir model of Eq. (4.3.1); however, a form of model is implicit in the ANN within the distribution of its weights. Moreover, this distribution is obtained automatically with no user intervention. Since the ANN works with total rainfalls and total flows, or changes in flows, there is no necessity to apply loss functions, base-flow separation and other such more-or-less (or illusory) physics-based techniques, as in conventional approaches. The ANN is indeed the ultimate hydrological black-box. The latency of the model appears, on the one hand, to be one of its greatest virtues; however, on the other hand, it may create even more dangerous situations, since the model now becomes, so to say, a 'prisoner' of its training data.

The numerical experiments reported here constitute, of course, only some first steps towards the testing of the generality of ANNs for use on more complex, real-world catchments, since all the problems of spatial distribution of rainfall and seasonal changes in catchment response have been avoided. However, promising results in the related field of distributed rainfall modelling have already been achieved by French *et al* (1992) in their study of the application of an ANN to forecast the rainfall over a grid of 25 x 25 points at time $t + 1$ from that at time t, using a network with 625 input and 625 output nodes.

The potential role of ANN models in hydrological modelling in general is manifold. At the simplest level, an ANN may function as a flexible, easy-to-implement, lumped-conceptual model that relates rainfall data to runoff data for individual catchments. At the other end of the spectrum, ANNs may be used to generate important components of physically-based, distributed hydrological modelling systems, whereby the ANN is used to induce a sub-model of individual physical processes (e.g. unsaturated zone flow dynamics) based only upon measured data (*see, for example,* Schaap and Bouten, 1996). Such a sub-model may then replace whole systems of complex, non-linear, differential equations that otherwise often require great skills from the modeller to calibrate and powerful computing devices to solve.

The greatest potential for artificial neural networks in hydroinformatics, however, would appear to be in the area of real-time forecasting and control in the general field of water resources management. The 'training-range' limitation may indeed be considered to be a serious restriction when dealing with measured data only, which is a restriction that applies for any 'black-box' method. However, by using results of physically-based, deterministic models to create 'off-line' training data sets it would become possible to create computationally-fast ANN models that cover all of the expected ranges of behaviour of the natural system, as demonstrated by Masood-Ul-Hassan *et al* (1995) in their study of the application of ANNs to the problem of real-time control of hydraulic structures in Bangladesh.

The results in Chapter 5 indicated that even if the exact mathematical formulation of a physical process is known, the use of ANNs in the solution of these partial and ordinary differential equations may offer an improvement over the traditional methods of numerical analysis, such as finite difference methods, which are plagued by problems of stability and accuracy. It has been shown that, in the simplest case of pure advection with a constant velocity, a linear ANN is capable of learning the exact solution, which is also exactly equivalent to the differential equation description. For problems of variable velocity, non-linear ANNs, i.e. ANNs with sigmoid threshold functions, have been shown to provide exceptionally accurate solutions over the range of velocities for which they were trained. In some cases, however, the ANN model was found to suffer from the problem of mass falsification in the solution. This problem may be seen as being equivalent to the problem of obtaining 'negative concentrations' when applying traditional numerical methods. The problem of mass falsification in ANN models can, however, be minimalised by ensuring that the training error has been reduced as close as possible to an absolute global minimum.

Finally, Chapter 6 demonstrated that ANNs may also be used as a data-mining technique to discover usable relationships in measured or experimental data. A common, more traditional approach to the analysis of measured and experimental data is through dimensional analysis and statistical curve fitting. Generally, the objective of such an analysis is simply to relate quantities that are very difficult to measure outside a specialised laboratory to parameters that can be easily measured in the field. Although the empirical formulae thus derived often fit the experimental data to a high degree of accuracy, these formulae often present the aspect of extremely complex, non-linear combinations of parameters and constants that do not really give much insight into the physical system being described. Also, the form and accuracy of the formulae are often very sensitive to the choice of parameters, dimensionless or otherwise. In many cases, for the sake of simplicity, several parameters, and hence measured data, may be disregarded entirely, at the cost of some accuracy in the final formulation of the relationship. The fact that the exact form of the empirical relation is thus not as important as the ability of the formula to map the experimental data accurately indicates that this kind of analysis may be very efficiently carried out using ANNs.

The true strength of the ANN paradigm lies in its ability to identify relationships between measured data without requiring a detailed knowledge of physical process characteristics *a priori*. The ANN is indeed a 'very black' box, where the user of the model has very little (if any) influence upon the form of model to be fitted to the measured data. The ANN does not explicitly identify a form of model but this form is implicit in the ANN, being encoded within the

distribution of weights. With traditional conceptual modelling techniques the modeller applies his or her measured data together with some physical insight in order to adjust modelling parameters and equations manually and so eventually to calibrate the model. In ANN modelling, one could almost speak of an *automatic calibration* procedure.

This superior performance characteristic of the ANN paradigm over the more traditional, manual methods of data mining and analysis can also be claimed by other sub-symbolic paradigms, such as genetic programming (GP). Although essentially sub-symbolic at its most basic level, GP will supply a symbolic-algebraic relation between the measured data through a process of evolution and competition between all possible solution expressions. An ANN, on the other hand, will usually find a relationship between the input and output data that has a much higher accuracy, but then the resulting relationships can only be represented sub-symbolically and are therefore essentially 'hidden' from the user of the model.

References

Abbott, M.B. (1969) Data-reversible systems for flood routing, *Proc. XIII Congress IAHR*, Kyoto, Japan, Vol. 1, IAHR, pp.305-312.

Abbott, M.B. (1979) Commercial and scientific aspects of applied mathematical modelling, *Advances in Engineering Software*, Vol. 1, No. 4, pp.147-152.

Abbott, M.B. (1979/1992) *Computational Hydraulics - Elements of the Theory of Free Surface Flows*, Ashgate, Aldershot, U.K.

Abbott, M.B. (1991) *Hydroinformatics - Information Technology and the Aquatic Environment*, Avebury Technical, Aldershot, U.K.

Abbott, M.B. (1992) The theory of the hydrologic model, or: The struggle for the soul of hydrology, in *Topics in Theoretical Hydrology: A Tribute to Jim Dooge*, O'Kane, J.P. (ed.), Elsevier, Amsterdam, pp. 237-259.

Abbott, M.B. (1993) The electronic encapsulation of knowledge in hydraulics, hydrology and water resources, *Advances in Water Resources*, No. 16, pp. 21-39.

Abbott, M.B. (1994) Hydroinformatics: A Copernican revolution in hydraulics, *Journal of Hydraulics Research*, Vol. 32, Extra Issue, IAHR, pp. 3-13.

Abbott, M.B., Bathurst, J.C., Cunge, J.A., O'Connell, P.E. and Rasmussen, J. (1986) An introduction to the European Hydrological System - Système Hydrologique Européen, "SHE", 2: Structure of a physically-based, distributed modelling system, *Journal of Hydrology*, No. 87, pp. 61-77.

Abbott, M.B. and Minns, A.W. (1997) *Computational Hydraulics*, 2nd Ed., Ashgate, Aldershot, U.K.

Amorocho, J. and Hart, W.E. (1964) A critique of current methods in hydrologic systems investigation, *Trans. Amer. Geophys. Union*, No. 45, pp. 307-321.

Askew, A.J. (1970) Derivation of formulae for variable lag time, *Journal of Hydrology*, Vol. 10, pp. 225-242.

Babović, V. (1993) Evolutionary algorithms as a theme in water resources, in *Scientific Presentations of AIO Meeting '93: AIO Network Hydrology*, Boekelman, R.H. (ed.) Delft University of Technology, The Netherlands, pp. 21-36.

Babović, V. (1996) *Emergence, Evolution, Intelligence; Hydroinformatics*, Ph.D Thesis, Balkema, Rotterdam.

Babović, V and Abbott, M.B. (1997) The evolution of equations from hydraulic data, *Journal of Hydraulics Research*, Vol. 35, No. 3, IAHR, pp. 397-430.

Babović V. and Baretta, J. (1996) Individual-based modelling of aquatic populations, *Hydroinformatics '96: Proc. 2nd International Conference on Hydroinformatics*, ETH-Zürich, Balkema, Rotterdam. Vol. 2, pp. 771-778.

Babović, V. and Minns, A.W. (1994) Use of computational adaptive methodologies in hydroinformatics, in *Hydroinformatics '94, Proc. 1st International Conference on Hydroinformatics*, Verwey *et al* (eds.), Balkema, Rotterdam, pp. 201-210.

Barth, K. (1932//1975) *Church Dogmatics, Vol. I; The Doctrine of the Word of God, Part 1*, translated from German by Bromiley, G.W., T. & T. Clark, Edinburgh.

Beale, R, and Jackson, T. (1990) *Neural Computing: An Introduction*, Institute of Physics Publishing, Bristol.

Boon, J.P. and Noullez, A. (1987) Lattice gas hydrodynamics, *Special Course on Modern Theoretical and Experimental Approaches to Turbulent Flow Structure and its Modelling*, AGARD Report No. 755, NATO.

Brillouin, L. (1956) *Science and Information Theory*, Academic Press, New York.

Cañizares, R., Heemink, A.W. and Vested, H.J. (1996) Sequential data assimilation in a fully non-linear hydrodynamic model, *Hydroinformatics '96, Proc. 2nd International Conference on Hydroinformatics*, ETH-Zürich, Balkema, Rotterdam, pp. 463-470.

Cunge, J.A. (1969) On the subject of flood propagation computation method (Muskingum Method), *Journal of Hydraulic Research*, Vol. 7, No. 2, IAHR, pp. 205-230.

Cunge, J.A., Holly, F.M. and Verwey, A. (1980) *Practical Aspects of Computational River Hydraulics*, Pitman, London, Reprinted by Institute of Hydraulic Research, Iowa.

Darwin, C. (1859) *The Origin of Species by Means of Natural Selection*, 6th ed., John Murray, London.

Denbigh, K.G. (1978) The objectivity, or otherwise, of the present, *The Study of Time III, Proc. 3rd Conference of the International Society for the Study of Time*, July, 1976, Alpbach, Austria, Springer-Verlag, New York, pp. 307-329.

Dibike, Y.B. (1997) *Use of Artificial Neural Networks in Model Approximation*, MSc Thesis HH 309, IHE-Delft

Diskin, M. H. and Simon, E. (1977) A procedure for the selection of objective functions for hydrologic simulation models, *Journal of Hydrology*, No. 34, pp. 129-149.

Donald, J.H. (1994) Rule induction - machine learning techniques, *Computing and Control Engineering Journal*, October, pp. 249-255

Einstein, H.A. (1950), The bed-load function for sediment transport in open channel flow, *Technical Bulletin* No. 1026, U.S. Dept. of Agriculture, Washington, D.C.

Engelund, F. and Fredsoe, J. (1976), A sediment transport model for straight alluvial channels, *Nordic Hydrology*, 7, pp. 293-306.

Fayyad, U., Piatetsky-Shapiro, G. and Smyth P. (1996) From data mining to knowledge discovery in databases, *AI Magazine*, American Assoc. for Artificial Intelligence, Vol. 17, No. 3, pp. 37-54.

Fogel, L.J., Owens, A.J. and Walsh, M.J. (1966) *Artificial Intelligence through Simulated Evolution*, Ginn, Needham Heights.

Foucault, M. (1966//1970) *The Order of Things, An Archeology of the Human Sciences*, translated from French, Routledge, London.

French, M.N., Krajewski, W.F. and Cuykendall, R.R. (1992) Rainfall forecasting in space and time using a neural network, *Journal of Hydrology*, Vol. 137, pp. 1-31.

Frisch, U., Hasslacher, B. and Pomeau, Y. (1986) Lattice gas automata for the Navier Stokes equations, *Physical Review Letters*, Vol. 56, No. 14, pp. 1505-1508 (reprinted in Wolfram, 1986).

Goles, E. and Martinez, S. (1990) *Neural and Automata Networks - Dynamical Behaviour and Applications*, Kluwer Academic, Dordrecht.

Guy, H.P., Simons, D.B. and Richardson, E.V. (1966), Summary of alluvial channel data from flume experiments: 1956-61, *Professional Paper* 462-I, U.S. Geological Survey, Washington, D.C.

Hall, M.J. (1977) The effect of urbanisation on storm runoff from two catchment areas in North London, in: *Effects of Urbanisation and Industrialisation on the Hydrological regime and Water Quality, Proc. Amsterdam Symp.*, International Association of Scientific Hydrology, No. 123, pp. 144-152.

Hall, M.J. (1997) How well does your model fit your data?, unpublished manuscript.

Hall, M.J. and Minns, A.W. (1993) Rainfall-runoff modelling as a problem in artificial intelligence: experience with a neural network, *Proc 4th Nat. Hydrol. Symp.*, Cardiff, British Hydrological Society, London, pp. 5.51-5.57.

Harnad, S. (1990) The symbolic grounding problem, in *Emergent Computation*, Forrest, S. (ed), MIT Press, Cambridge, pp. 335-346.

Hebb, D.O. (1949) *The Organization of Behavior*, Wiley, New York.

Heidegger, M. (1927//1962) *Being and Time*, translated from German by Macquarrie, J. And Robinson, E., Blackwell, London.

Hertz, J., Krogh, A. and Palmer, R.G. (1991) *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City, CA.

Holland, J.H. (1975) *Adaption in Natural and Artificial Systems*, Univ. Of Michigan Press.

Holland, J.H. (1986) Escaping brittleness: The possibilities of general purpose learning algorithms applied to parallel rule-based systems, in *Machine Learning: An Artificial Intelligence Approach*, Michalski, R.S., Carbonell, J.G. and Mitchell, T.M. (eds.), Morgan Kaufmann, San Mateo.

Hong Kee An and Mohd. Nor Bin Hj. Mohd. Desa (1988) Application of runoff routing model to the Krian river basin. *Bull Instn. Engrs. Malaysia*, pp. 12-17.

Hopfield, J.J. (1982) Neural networks and physical systems with emergent collective computational abilities, *Proc. National Academy of Sciences USA*, No. 79, pp. 2554-2558.

Hopfield, J.J. (1994) Neurons, dynamics and computation. *Physics Today*, 47 (2): 40-46.

Hornik, K., Stinchcombe, M. and White, H. (1989) Multilayer feedforward networks are universal approximators, *Neural Networks*, 2: 359-366.

IAHR (1994) *Extra Issue, Journal of Hydraulic Research*, Vol 32, IAHR, Delft, 214pp.

Iba, H., de Garis, H., and Sato, T. (1993) *Solving Identification Problems by Structured Genetic Algorithms*, ETL-TR-93-17.

Izzard, C.F. (1946) Hydraulics of runoff from developed surfaces, *Proceedings of the Highway Research Board*, No. 26, pp. 129-146.

Izzard, C.F. and Augustine, M.T. (1943) Preliminary report on analysis of runoff resulting from simulated rainfall on a paved plot, *Transactions of the American Geophysical Union*, No. 24, pp. 500-509.

Johnstone, D. and Cross, W.P. (1949) *Elements of Applied Hydrology*, Ronald Press, New York, 275 pp.

Kalman, R.E. (1960) A new approach to linear filter and prediction theory, *Journal of Basic Engineering*, Vol. 82, pp. 35-45.

Koza, J. (1992) *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge.

Laurenson, E.M. (1964) A catchment storage model for runoff routing. *Journal of Hydrology*, No. 2: pp. 141-163.

Laurenson, E.M. and Mein, R.H. (1988) *RORB - Version 4 runoff routing program - user manual*, Monash Univ., Clayton, Aus.

Linsley, R.K. (1967) The relation between rainfall and runoff, *Journal of Hydrology*, No. 5, pp. 297-311.

Maksimovic, C. and Radojkovic, M. (1986) *Urban Drainage Catchments. Selected Worldwide Rainfall-Runoff Data from Experimental Catchments*. Pergamon Press, Oxford.

Mason, J.C., Price, R.K. and Tem'me, A. (1996) A neural network model of rainfall-runoff using radial basis functions, *Journal of Hydraulics Research*, Vol. 34, No. 4, IAHR, pp. 537-548.

Masood-Ul-Hassan, K., Minns, A.W. and Yde, L. (1995), Application of artificial neural network to real-time control of hydraulic structures, *Advances in Intelligent Data Analysis*, Lasker, G. and Liu, X. (eds), IIAS Press, pp. 114-118.

Maynard-Smith, J. (1973) *Models in Ecology*, Cambridge University Press, Cambridge.

McCulloch, W.S. and Pitts, W. (1943) A logical calculus of the ideas immanent in nervous activity, *Bulletin of Mathematical Biophysics*, No. 5, pp. 115-133.

Mein, R.G., Laurenson, E.M. and McMahon, T.A. (1974) Simple nonlinear model for flood estimation, *Proceedings of the American Society of Civil Engineers, Journal of the Hydraulics Division*, No. 100 (HY11), pp. 1507-1518.

Minns, A.W. (1995) Analysis of experimental data using artificial neural networks, *HYDRA 2000, Proc. XXVI Congress IAHR*, London, Vol. 1, Thomas Telford, London, pp. 218-223.

Minns, A.W. (1996) Extended rainfall-runoff modelling using artificial neural networks, *Hydroinformatics '96, Proc. 2nd International Conference on Hydroinformatics*, ETH-Zürich, Balkema, Rotterdam, pp. 207-213.

Minns, A.W. and Babović, V. (1991) Hydroinformatics: balancing the aquatic equation, *Euro Holland Trade*, No. 15, Informa, The Hague, pp. 28-31.

Minns, A.W. and Babović, V. (1996) Hydrological modelling in a hydroinformatics context, in *Distributed Hydrological Modelling*, Abbott, M.B. and Refsgaard, J.C. (eds.), Kluwer Academic, Dordrecht, pp. 297-312.

Minns, A.W. and Hall, M.J. (1996) Artificial neural networks as rainfall-runoff models, *Hydrological Sciences Journal*, Vol. 41, No. 3, pp. 399-417.

Minns, A.W. and Hall, M.J. (1997) Living with the ultimate black box: more on artificial neural networks. *Proc 6th Nat. Hydrol. Symp.*, Salford, British Hydrological Society, London, pp. 9.45-9.49

Minsky, M.L. and Papert, S.A. (1969/1988) *Perceptrons: An Introduction to Computational Geometry*, Expanded Edition, MIT Press, Cambridge, MA.

Natural Environment Research Council (1975) *Flood studies report, Vol II, Meteorological studies*, The Council, London.

Newell, A. (1980) Physical symbol systems, *Cognitive Science*, Vol. 4, pp. 135-183.

Park, D. (1978) The past and the future, *The Study of Time III, Proc. 3rd Conference of the International Society for the Study of Time*, July, 1976, Alpbach, Austria, Springer-Verlag, New York, pp. 351-367.

Refsgaard, J.C., Seth, S.M., Bathurst, J.C., Erlich, M., Storm, B., Jørgensen, G.H. and Chandra, S. (1992) Application of SHE to catchments in India, Part 1. General results, *Journal of Hydrology*, No. 140, pp. 1-23.

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

Rumelhart, D.E., McClelland, J.L. and the PDP Research Group (1986) *Parallel Distributed Processing - Explorations in the Microstructure of Cognition*, 2 Volumes, MIT Press, Cambridge, MA.

Rumelhart, D.E., Widrow, B. and Lehr, M.A. (1994) The basic ideas in neural networks, *Communications of the ACM*, 37 (3): 87-92.

Savenije, H.H.G. (1992) *Rapid Assessment Technique for Salt Intrusion in Alluvial Estuaries*, Ph.D. thesis, IHE Report Series 27, IHE, Delft, The Netherlands

Savenije, H.H.G. (1993a) Composition and driving mechanisms of longitudinal tidal average salinity dispersion in estuaries, *Journal of Hydrology*, 144, pp. 127-141.

Savenije, H.H.G. (1993b) Predictive model for salt intrusion in estuaries, *Journal of Hydrology*, 148, pp. 203-218.

Schaap, M.G. and Bouten, W. (1996) Modelling water retention curves of sandy soils using neural networks, *Water Resources Research*, Vol. 32, pp. 3033-3040.

Schwefel, H.-P. (1981) *Numerical Optimisation of Computer Models*, Wiley, Chichester.

Selvalingham, S., Liong, S.Y. and Manoharan, P.C. (1987) Application of RORB model to a catchment in Singapore. *Wat. Resour. Bull.*, Vol. 23, No. 1, pp. 81-90.

Shannon, C.E. and Weaver, W. (1949) *The Mathematical Theory of Information*, U. of Illinois Press, Urbana, Ill.

Shawkat Ali, Md. (1997) *Applications of Self Organizing Feature Maps for the Analysis of Hydrological and Ecological Data Sets*, MSc Thesis HH 315, IHE-Delft.

Smith, M. (1993) *Neural Networks for Statistical Modelling*, Van Nostrand Reinhold, New York.

Smith, J. and Eli, R.N. (1995) Neural network models of rainfall-runoff process, *Journal of Water Resources Planning and Management*, Vol. 121, No. 6, pp. 499-508.

Smolensky, P. (1988) On the proper treatment of connectionism, *Behavioral and Brain Sciences*, No. 11, pp. 1-74.

Szilard, L. (1929) *Zsch. f. Physik*, Vol. 53, p. 840.

Tang, Z and Fishwick, P.A. (1993) Feedforward neural nets as models for time series forecasting, *ORSA Journal of Computing*, Vol. 5, No. 4, pp. 374-385.

Verwey, A. and Ilić, S. (1993) A space-compact high-order implicit scheme for 1-D advection simulations, *Proc. XXV Congress IAHR*, Tokyo, Vol. V, pp. 355-362.

Verwey, A., Minns, A.W., Babović, V. And Maksimović, C. (eds.) (1994) *Hydroinformatics '94. Proc. 1st International Conference on* Hydroinformatics, IHE-Delft, 2 Vols, Balkema, Rotterdam.

Von Neumann, J. (1949) Theory of self-reproducing automata, *Univ. of Illinois Lectures on the Theory and Organization of Complicated Automata*, Burks, A.W. (ed), Univ. of Illinois Press, Urbana.

Walker R., Gerrets, M. And Haasdijk, E. (1993), *A Genetic Algorithm for the Approximation of Formulae*, ESPRIT Project 6857, Deliverable D2.2.

Watkins, L.H. (1962) The design of urban drainage systems. Research into the relation between the rate of rainfall and the rate of flow in sewers, *Road Research Technical Paper*, No 55, HMSO, London.

Watt, W.E. and Kidd, C.H.R. (1975) QUALRM - a realistic urban runoff model, *Journal of Hydrology*, No. 27, pp. 225-235.

Widrow, B. and Hoff, M.E. (1960) Adaptive switching circuits, *Institute of Radio Engineers, WESCON Convention Record*, Part 4, pp. 96-104.

Wilson, G. (1988) The life and times of cellular automata, *New Scientist*, No. 1633.

Wilson, S.W. (1994) ZCS: A zeroth level classifier system, *Evolutionary Computation*, Vol. 2, No. 1, pp. 1-18.

Wittgenstein, L. (1953//1992) *Philosophical Investigations*, 3rd ed., translation from German by Anscombe, G.E.M., Blackwell, Oxford.

Wolfram, S. (1986) *Theory and Applications of Cellular Automata*, World Scientific, New York.

Wuensche, A. and Lesser, M. (1992) *The Global Dynamics of Cellular Automata: An Atlas of Basin of Attraction Fields of One-Dimensional Cellular Automata*, Reference Volume I, Santa Fe Institute studies in the sciences of complexity, Addison-Wesley, Reading, MA.

Yu, Y.S. and McNown, J.S. (1964) Runoff from impervious surfaces. *Journal of Hydraulic Research*, Vol 2, IAHR, pp. 3-24.

Zyserman, J.A. and Fredsøe, J. (1994), Data analysis of bed concentration of suspended sediment, *Journal of Hydraulic Engineering*, ASCE, Vol. 120, No. 9, pp. 1021-1042.

Appendix: Overfitting and Generalisation

The back-propagation algorithm is an extremely efficient method for fitting a function to any given set of data points. However, in some circumstances it may end up fitting a function far too well. That is, rather than finding a function that describes the general underlying relationship that links the data points, we may end up with a function that actually passes exactly through each data point, and may thus include all of the 'noisy' data points as well as any outliers in the data set. This phenomenon is referred to as *overfitting*. Smith (1993) provides a particularly detailed discussion of this problem. A simple example will suffice here to demonstrate the problem as it applies to learning in artificial neural networks.

Consider a set of data scattered about the straight line $y = x$ as sketched in Fig A.1. A least-squares, linear regression of the data would lead to the fitting of the straight line function $y = x$ as indicated in Fig. A.1.

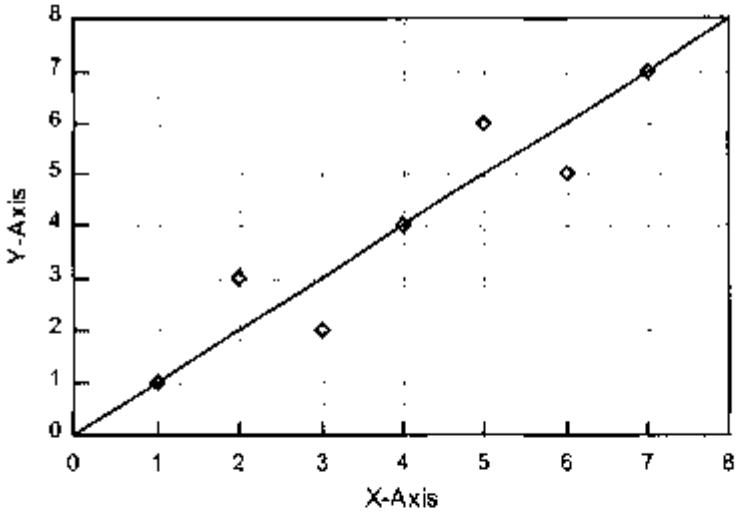


Fig A.1 Example data set scattered about the straight line $y = x$

However, by using an artificial neural network, or for that matter any other non-linear regression technique, it is also possible to fit a higher-order function to the data as sketched in Fig. A.2.

Any measure of the error between the higher-order function and the data points in Fig. A.2 would obviously be much less than the error between the linear function and the data points in Fig. A.1. However, the higher-order function does not necessarily provide a better solution to the problem in this case. It has not *generalised* the relationship between x and y, and will in fact perform very

poorly when used to interpolate the value of $y = x$ for values that lie between the given data points.

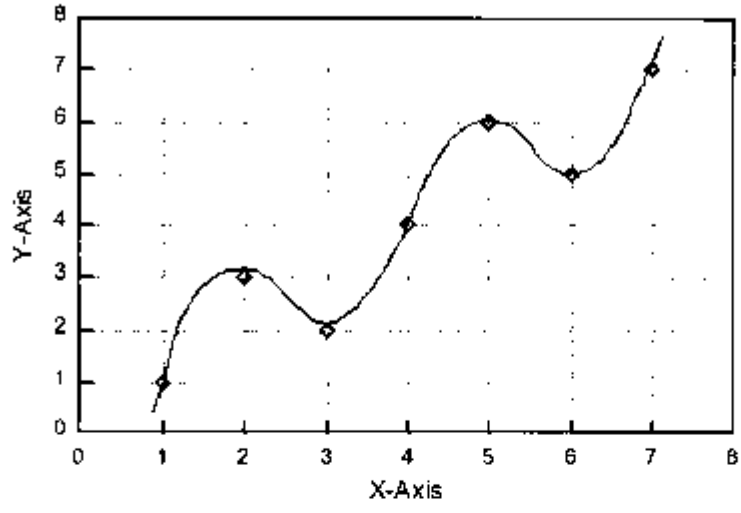


Fig A.2 Example of overfitting a function to the data

This simple example suffices to show the results of overfitting and indicates that the only way to check whether overfitting has occurred is either to inspect the fitted curve visually, or to test the fitted function with other data points which lie in between the original data points. This latter procedure is the recommended procedure for the training and verification of an ANN. It is vital that, after a network has been trained, it is confronted with a new set of data that it has not 'seen' before. If the performance of the ANN on this verification data set differs significantly from the performance of the ANN on its own training data set, then it can be concluded that the ANN has not generalised the training data and the solution is in fact less satisfactory, and possibly actually worthless.

The shape of the function fitted by an ANN to a set of training data points will depend very strongly upon the degrees of freedom provided by the network structure. The number of degrees of freedom is very closely linked to the total number of nodes and connections in the network. An increase in the number of hidden nodes in an ANN will increase the number of degrees of freedom and thus increase the capacity of the network to model more complex functions accurately. It will also, however, increase still further the capacity of the network to overfit.

The effect of changing the number of hidden nodes in an ANN can most easily be demonstrated by using to the example data given above. The data set depicted in Fig. A.1 was used to train four different ANNs with 1, 2, 3 and 4 hidden nodes respectively. In each case, the ANN was trained as long as possible until there were no more changes occurring in the weight configurations. Each configuration was also trained several times with different initial random distributions of weights to ensure that the network had converged on the global minimum error. The results of the training procedure are depicted in Figs. A.3a and A.3b.

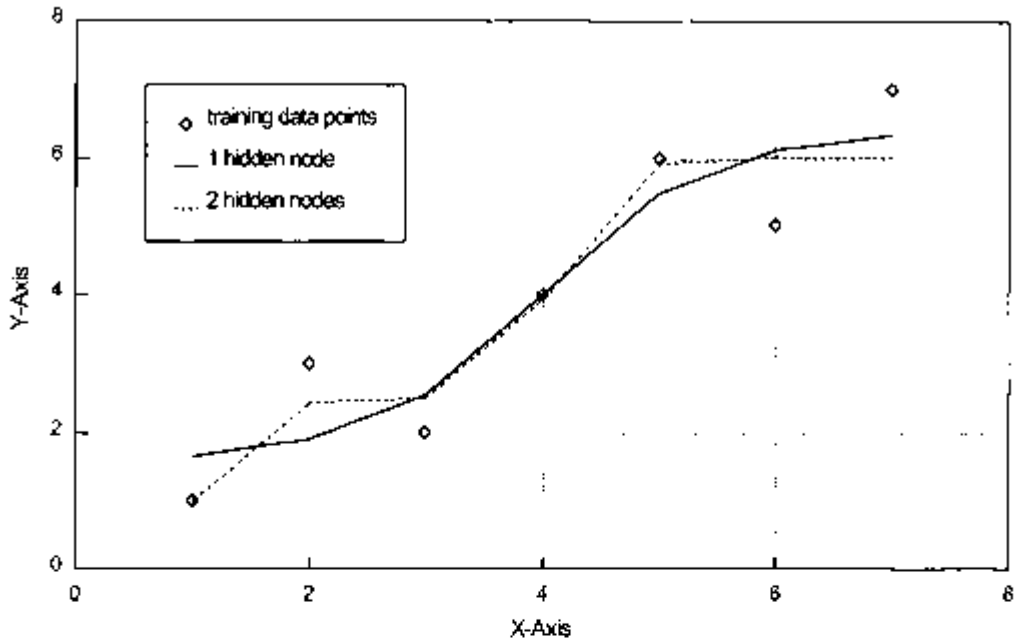


Fig. A.3a Results of training an ANN using 1 and 2 hidden nodes

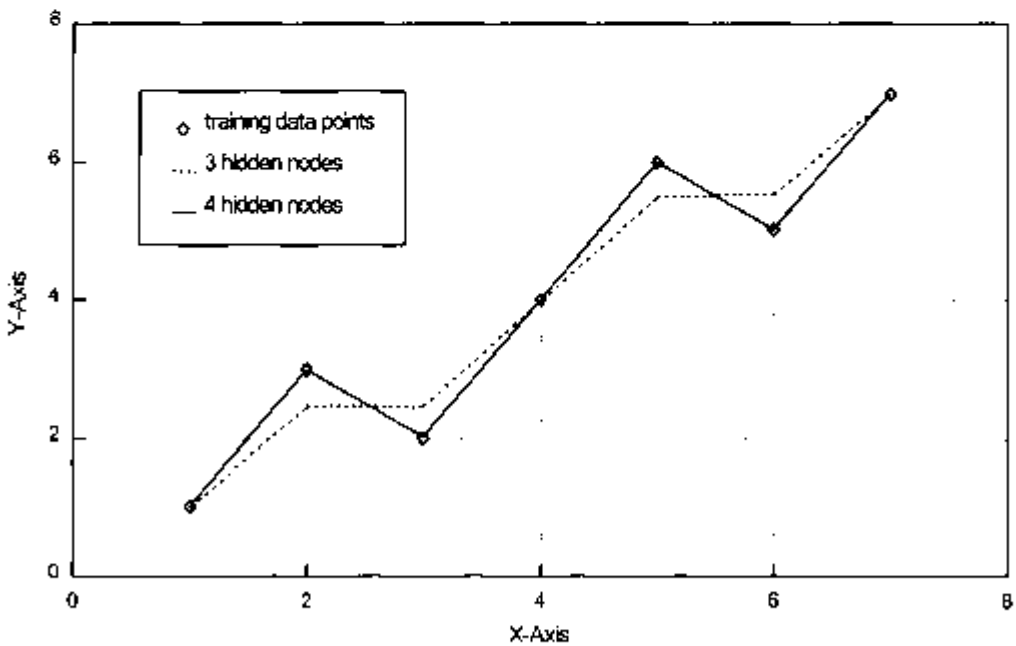


Fig. A.3b Results of training an ANN using 3 and 4 hidden nodes

The global minimum errors for each ANN configuration, expressed as the residual RMS error between the fitted function and the training data, are given in Table A.1.

Table A.1 RMS errors obtained after training ANNs with varying numbers of hidden nodes

Number of hidden nodes	RMS error on training data set
1	0.753
2	0.604
3	0.385
4	0.006

From these results it is obvious that the performance of an ANN can be significantly influenced by the choice of the number of hidden nodes. Although the network with four hidden nodes had the least residual error after training, the network with only one hidden node learned a function that provided a much more generalised description of the training data.

Many of these problems of overfitting and generalisation can be overcome by increasing the sample size of the training data set. If the training data set contains a sufficiently large number of data points, the ANN will not be able to pass a function exactly through each data point and it will be forced to generalise. For example, if there were more data available in the simple example described above, it would become clear whether the scatter of data points about the line $y = x$ was merely due to noise or whether the variation was in fact due to some other underlying relationship between x and y.

Throughout this study, and especially for the problem of rainfall-runoff modelling described in Chapter 4, the data sets were extremely large (e.g. several thousands of data points in the training and verification data sequences) and the problem of overfitting did not generally occur. For these types of applications, it is usually sufficient to find a network configuration with the minimum number of hidden nodes that can successfully learn to fit the given training data. Above a certain number of hidden nodes the training performance will not be significantly improved by the addition of further nodes.

However, overfitting may also occur due to *overtraining* of the network. As the network is trained, its mapping function grows more and more complex. At some point it will arrive at a configuration that gives the best generalisation, after which any additional learning will simply give rise to overfitting.

The effect of the length of the training time upon the performance of the network can again be demonstrated using the simple example above. In this case, a network with four hidden nodes was trained once more on the given data, but the training was stopped at carefully chosen times during the training process. The results of the training at the intermediate time levels are depicted in Figs. A.4a and A.4b.

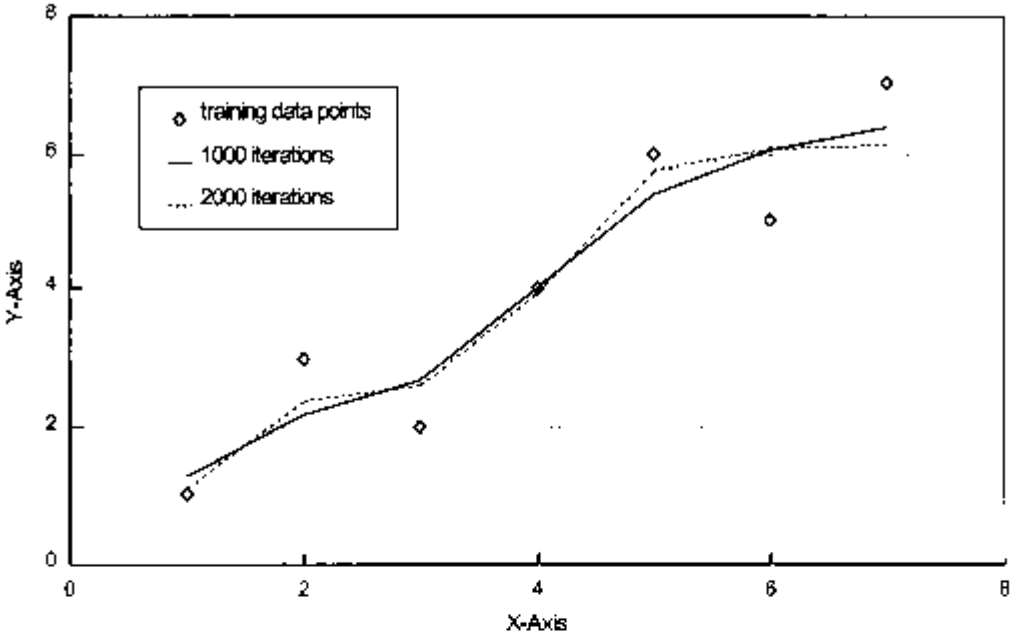


Fig A.4a Intermediate results obtained by stopping the training of an ANN with 4 hidden nodes after 1000 and 2000 iterations on the training data set

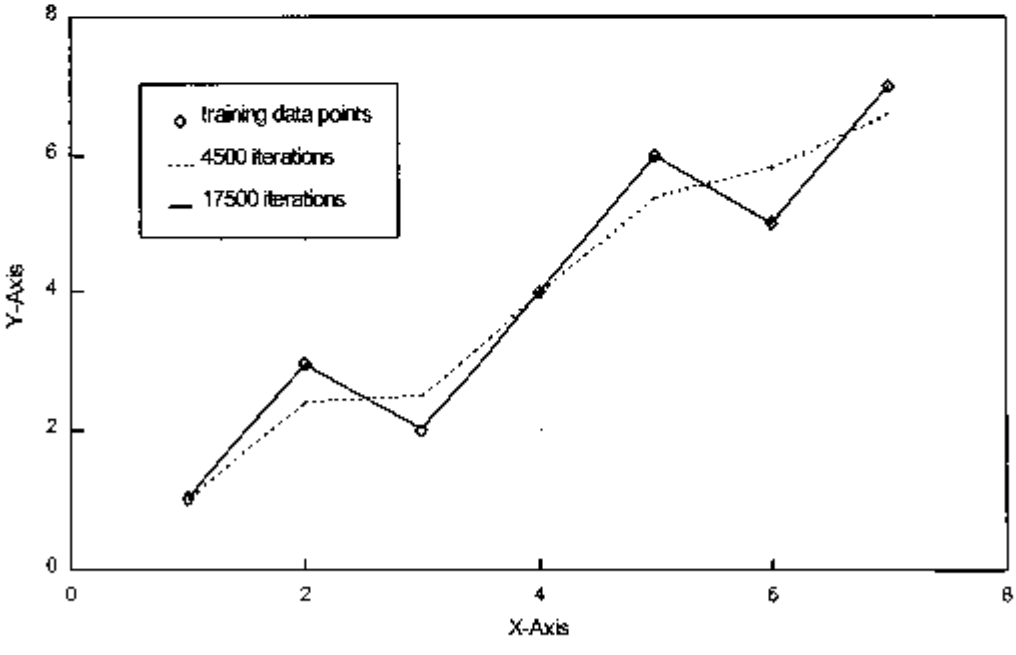


Fig A.4b Intermediate results obtained by stopping the training of an ANN with 4 hidden nodes after 4500 iterations on the training data set compared to the final convergence obtained after 17500 iterations

A comparison between Figs. A.3 and A.4 shows a striking similarity between Figs. A3a and A4a and between Figs A3b and A4b. The results from the converged network with one hidden node are almost identical to the intermediate training results from the ANN with 4 hidden nodes after 1000 iterations. Similarly, we obtain comparable mapping functions between the converged networks with two and three hidden nodes and the intermediate training results after 2000 and 4500 iterations respectively. The ANN with four hidden nodes finally converges to the least-RMS-error solution after 17500 iterations.

During the training process, the network appears to pass through successive stages in which the number of hidden nodes that are actually contributing to the solution goes up one by one. Consequently, the network passes through stages during which its output is similar to the output of converged networks with various numbers of hidden nodes.

Smith (1993) suggests a procedure to check for overfitting by controlling the error on the validation data sequence as follows:

1. Divide the data set into training and validation sub-sets.
2. Train the network on the training data set.
3. Periodically stop the training and measure the error on the validation data set. Save the weights of the network before continuing.
4. Repeat steps 2 and 3 until the error on the validation data set starts to increase. This is the moment that overfitting has begun. Stop the training of the network and go back to the weights that produced the lowest error on the validation data set, and use these weights for the trained ANN model.

This procedure also enables us to avoid the problem of finding the exact number of hidden nodes to give the best generalisation properties. By following steps 1 to 4, we can simply configure a network that has at least enough hidden nodes to learn a relationship from the data set, and consequently stop the training process at the right time.

As mentioned earlier, the large amounts of data used in the experiments in Chapter 4, meant that overfitting was generally not a problem. The networks were trained for very long periods of time in order to reduce the RMS error on the training data set, and this almost always went together with a subsequent decrease in the error on the validation data set. Only in one or two cases did the effects of overlearning become apparent after a very, very long time. Since the problem of overfitting was considered to be negligible, the problem of the selection of the number of hidden nodes was then simply reduced to finding the minimum number of nodes that would still give satisfactory results. This then led to the selection of the smallest possible networks in each case, which were subsequently quicker and easier to train.

IHE
DELFT

The aim of the International Institute for Infrastructural, Hydraulic and Environmental Engineering, IHE Delft, is the development and transfer of scientific knowledge and technological know-how in the fields of transport, water and the environment.

Therefore, IHE organizes regular 12 and 18 month postgraduate courses which lead to a Masters Degree. IHE also has a PhD-programme based on research, which can be executed partly in the home country. Moreover, IHE organizes short tailor-made and regular non-degree courses in The Netherlands as well as abroad, and takes part in projects in various countries to develop local educational training and research facilities.

International Institute for
Infrastructural, Hydraulic and
Environmental Engineering

P.O. Box 3015
2601 DA Delft
The Netherlands

Tel.: +31 15 2151715
Fax: +31 15 2122921
E-mail: ihe@ihe.nl
Internet: http://www.ihe.nl

Hydroinformatics proceeds into neural networks [illegible]. Abiotic life characterised as the most symbiotic [illegible] two different paths. Along the one path, elaborate tools and even more general working environments for engineers, environmentalists and other professionals that make little or no use of symbols. In any forward direction, but which proceed almost entirely without signs. Along the other path, as illustrated in this present work, hydroinformatics increasingly uses new symbolic and mental activity sub-symbolic methods in order to design these tools and more general working environments. Of course, in this latter case, the application of these environments must still make recourse to symbolic processes, but, as explained here, these are relegated eventually as aids to the thinking processes of the constructor, and are not [illegible] over or incorporated in any way into the operations of the system [illegible]. It is this second path of sub-symbolic constructions that forms the subject of the present work.

The three main [illegible] sub-symbolic paradigm and its [illegible] networks, evolutionary algorithms and cellular automata. Artificial neural networks and, in particular, feed forward multi-layer perceptrons, constitute the only symbolic tool used throughout this work.

IHE DELFT