Regardless of the size of your computer system it is a strategic tool. We are committed to helping you use it most effectively. All it needs is a little imagination.

UNIDATA
Telephone (012) 45-1152

THERE'S A GIANT IN EVERY ONE OF US
The Department of Computer Science at Pretoria University is inviting applications for two annually awarded bursaries for MSc study. These bursaries are aimed at outstanding students who have a Computer Science Honours degree (or equivalent qualification). As far as possible, one such bursary will be awarded to a South African student, while the other will be awarded to a student from another African state. The hope is to thereby promote interaction between South African computer scientists and computer scientists elsewhere in Africa.

The bursaries are funded by the Department's contract research work and will amount to a minimum of R20000 per year. They are intended to cover costs for tuition, housing, transport, books, and an adequate allowance for other living expenses. Successful candidates will be required to work for 20 hours per week as part of the Computer Science Department's contract research team. The work assigned to each candidate will be aimed at providing him with practical experience in his field of interest.

Study fields for which bursaries will be considered include:
- Computer Networking
- Distributed Systems
- Software Engineering
- Artificial Intelligence
- Intelligent Computer Aided Instruction
- Machine Aided Translation

Students should aim to complete the dissertation that is required for an MSc within two years of study (i.e. one year of full time study). The dissertation should preferably be written in English but it may also be written in Afrikaans.

The continuation of the bursary and adjustments to the allowance will be decided upon annually and will depend on progress. It is a condition of the bursary that, after completing his/her studies, the student should return to his/her country of origin for the same number of years for which the bursary was held.

Applicants may request application forms from:

MSc. Bursaries
The Computer Science Department
University of Pretoria
Hillcrest
Pretoria
0083
South Africa
Editor's Notes

It is with sincere gratitude that SACJ takes leave of Dr Peter Lay who, until recently, was the assistant editor dealing with Information Systems. He has left academia for what sounds like a more gentle lifestyle. (He has gone farming!) Under Peter's stewardship the number of high-quality IS papers in SACJ grew steadily. In general, IS papers tend to be accessible and relevant to a wide spectrum of computer professionals, and the quality of IS papers that have been appearing in SACJ has significantly contributed to the increased interest being shown in the journal by the local computer industry. If this growth in interest is to be sustained, it is urgent and important to find a suitable replacement assistant editor. The ideal candidate should not only be respected as an academic by his peers, but should also be disposed to enthusiastically promote SACJ in the private sector. Since a shortlist of candidates is currently being compiled, I would like issue a general appeal for names that might be included on it. Please contact me urgently if you would like to be considered for the job, or if you would like to nominate someone that you consider to be particularly suitable.

My three year term of office as editor expires in October. I have always considered it a great privilege to hold this position, and as a result, I felt honoured when the SAICS executive committee requested that I stay on for a further term. Nevertheless, I initially declined the request on the grounds that the time-demands of the job were significantly eroding my ability to fulfil other duties. Particularly demanding has been the task of seeing to the typesetting of the various contributions - either by doing it myself, or by ensuring that it is adequately done by someone else. Recently, however, Prof G de V Smit (Riel Smit) at UCT has offered to assume the role of production editor. This generous offer so much changes the complexion of what is being asked of me that I am now both willing and honoured to continue as editor for another term. I am very grateful to Riel for his offer and I look forward to working with him. In future, authors whose papers have been accepted for publication will be asked to liaise directly with him regarding the precise form in which the final contribution should be submitted.

The next issue of SACJ will consist largely of a selection of papers that were presented at the 6th South African Computer symposium. The selection will be based on comments from the referees who, at the time, were asked to adjudicate the papers in terms of their appropriateness for both the conference as well as for SACJ publication. Papers which, in the opinion of one or more referees, required major revision will have to be resubmitted to SACJ for refereeing purposes. Authors will soon be contact in this regard.

At the time of writing, the updated list of "approved" publications for the first half of 1991 had not yet been released by the relevant authorities. For the sake of past, present and future contributors I sincerely hope that SACJ will be on the list when it eventually comes out. However, I have become increasingly aware that there is a real danger of laying too much store on papers published in so-called approved journals as a basis for evaluating and rewarding research. I hope to expand more fully on this theme in a future edition of SACJ. Keep watching this space!

Derrick Kourie
Editor

This SACJ issue is sponsored by
a generous donation from
UNIDATA
Why the Fuss About Neural Networks?

Gerhard Barth

German Research Center for Artificial Intelligence, Kaiserslautern, Saarbrucken

Introduction

Currently, there is a certain hype about issues related to neural networks. This is somewhat surprising since the origins of neural nets date back some 40 years. At that time, several researchers experimented with distributed systems composed of so-called perceptrons. Problems tackled therewith included computer vision, weather prediction, speech recognition, cardiographic diagnosis and image classification. Chapter 2 of reference [1] contains more details.

This short contribution is intended to summarize in a very succinct way the main characteristics of neural nets as currently used. In particular, the two dominating training paradigms are briefly described. An attempt is made to sketch some guidelines as to what kinds of problems are good candidates for being tackled by neural networks.

The Basic Model of Computation

Neural networks are patterned after the most sophisticated and powerful problem solving device ever built - the human brain. The latter is a huge network of processing elements called neurons. They exchange signals through synaptic links. Please consult [1] for more information on biological and physiological issues. Figure 1 gives some more information about the number of neurons, their degree of connectedness and the speed of individual units. Although a single neuron may send off impulses to its neighbours at a modest rate only, the overall computational power of the entire network is extraordinary. Because neurons can operate in parallel, the human brain is able to perform some $10^{15}$ to $10^{18}$ operations during each and every second! In contrast, currently used neural networks are pretty small-scale systems. Most of them are built in software and run on conventional computers. Consequently, there are orders and orders and orders of magnitude difference between the capacities of human brains and what will be called neural network in the remainder of this paper.

Figure 1

Figure 2 depicts the way in which neurons are modeled. Each unit is equipped with a finite number of input ports through which signals may be received. Input channels are associated with weight factors, indicating their throughput capacity. In most practical applications, both input signals, $i_n$, and weights, $w_n$, are represented by integer or real numbers. The first step performed by a neuron is the calculation of the sum of all products $i_n * w_n$. The result thereof is used as argument for a function which describes the activity of the neuron. This mapping is normally performed by a threshold function which produces a non-zero output only if its argument exceeds some predefined value. Since step-functions possess some undesirable mathematical characteristics, for most practical applications continuous approximations are used. The usefulness thereof will become more obvious in the context of the error backpropagation training method.
Neurons as described above can be arranged in almost any way to create networks. Normally, layered architectures are used in practice. Whereas in the early days of neural network computing single-layer systems were popular, most recent applications are based on architectures with several layers. The top and bottom layer are commonly referred to as input and output layers, respectively. The layers in between contain hidden neurons. Typically one to three hidden layers exist in practical applications.

**Construction of Neural Nets**

The lifetime of a neural network can be split in two phases. Firstly, it is trained. To this end it is confronted with sample data and adjusted to the proper processing of them. Having done this for sufficiently many test patterns, the net is switched into the work mode. Then, problem-specific data are propagated from input neurons through hidden layers towards output neurons. Figure 3 illustrates this organization. Moreover, it indicates that at any moment the net may be reset into the training mode. It is the duty of the developer to switch among these two different states. In addition to that, he is responsible for many more aspects, like:

- finding the proper net topology;
- choosing the right training paradigm; and
- picking an appropriate set of test data.

Most training methods used for neural networks are modifications or extensions of the delta learning rule. Its basic idea is sketched in Figure 4.

The adjustment of weights is done by the following steps:

```plaintext
for all pairs (in, res) of test data do
    out := \text{Activation}(in \cdot w)
    error := res - out
    delta := \alpha \cdot error \cdot in
    w := w + \delta
```

A few things are worth mentioning about these statements. Firstly, if the output actually produced by the net matches the expected result, no adjustment of weights is done. Secondly, only neurons receiving a non-zero input are able to learn. Thirdly, the adjustment term delta is tuned by the factor, \( \alpha \), called learning rate. Typically, training starts with a value thereof which is close to 1. Thereafter it is gradually decreased, thus simulating a training principle where large adjustments of weights are followed by ever smaller ones. Finding the proper tuning of the learning rate is one of the more challenging problems in the construction of a neural network.

**Learning by Error Backpropagation**

For most real-life applications neural networks with more than one layer of processing elements are required. This means that signals are propagated from input neurons through hidden units to output neurons. It can be shown that this arrangement of neurons considerably improves the computational power of nets. Yet, training becomes more complicated. The delta rule cannot be applied directly to multi-layer architectures, because it requires a direct comparison between the expected result and the actual output of the net. To overcome these complications, multi-layer networks are trained according to the principle depicted in Figure 5. This involves two successive phases. As usual, input signals are sent from input towards output neurons. Having done this, the weights of output neurons can be updated by a slight modification of the delta rule. These adjustments are then propagated in a clever way through the hidden layers back to the input neurons. This strategy of pushing errors opposite to the direction of data flow through the network coined its name: error backpropagation.
Figure 6 sketches the training of output neurons. The following steps have to be performed:

\[
\begin{align*}
\text{error} & := \text{res} - \text{out} \\
\text{delta} & := \text{error} \cdot A'(\cdot) \\
\text{DELTA} & := a \cdot \text{delta} \cdot \text{in} \\
\text{w} & := \text{w} + \text{DELTA}
\end{align*}
\]

\(A'\) denotes the derivative of the activation function \(A\). Observe that its use causes larger adjustments in states, where the net to be trained operates at a high level of activation. The use of the derivative also directs adjustments towards minima of the values of delta and \(\text{DELTA}\), respectively. It happens more often than not that thereby the global minimum is missed, and training comes to a stop at a local minimum. To avoid such pitfalls, several remedies have been investigated. Consult references [1] and [4] for further details - the method called simulated annealing in particular.

Figure 7 in Figure 7 the training of non-output neurons is depicted. Remember that training proceeds in the opposite direction of data flow, from output through hidden towards input units. Consequently, when a non-output neuron is about to be trained, the weights of all neurons receiving its outgoing signals have already been adjusted. Moreover, their \(\text{delta}_1, ..., \text{delta}_n\) terms have been calculated. The following assignments specify the weight adjustments for non-output neurons:

\[
\begin{align*}
\text{delta} & := \Sigma (w_i \cdot \text{delta}_i) \\
\text{delta} & := \text{delta} \cdot A'(\cdot) \\
\text{DELTA} & := a \cdot \text{delta} \cdot \text{in} \\
\text{w} & := \text{w} + \text{DELTA}
\end{align*}
\]

The most interesting step thereof is the first one. It affects training of a neuron in proportion with its capacities to propagate potential errors to its immediate neighbours. The other steps of the training process do not differ from the treatment of output neurons.

**Competitive Learning Without a Trainer**

Learning by error backpropagation requires cooperation between the net to be trained and a teacher. The latter has to give feedback on the quality of the result computed by the net. This assumption does not always appropriately reflect reality. In many situations we are able to find concepts without being advised by someone. To illustrate the very basic idea of Kohonen nets let us concentrate on the special case of neurons having only two input channels. This allows us to represent them by pairs of weight values, i.e. they can be depicted as two-dimensional vectors in the Euclidean plane. Figure 8 contains such a graphical representation of four neurons and an input vector. To learn, competition among neuron arises. The one closest to the input vector is declared the winner. It is allowed to adjust its weight in such a way that it moves a little closer to the input vector. This learning principle can be somewhat formalized as follows:

\[
\begin{align*}
\text{WEIGHTS} & := \ldots \text{initialize} \ldots \\
\text{for all test data } t \text{ do} \\
\quad \text{for each neuron } n \text{ do} \\
\quad \quad \text{compute the distance between } t \text{ and } n \\
\quad \quad \text{winner} & := \text{neuron with the shortest distance} \\
\quad \quad \text{wwinner} & := \text{wwinner} + a \cdot (t - \text{wwinner})
\end{align*}
\]

As before, factor \(a\) denotes the learning rate, and decreasingly ranges between 1 and 0. Observe that no learning progress is made in cases where the input completely matches any of the Kohonen neurons.
Having presented sufficient test data, the net might have stabilized in a configuration such as shown in Figure 9. Clusters of neurons have been formed, each representing a concept that is sufficiently distinct from other ones. This could have happened without cooperation of a teacher knowing in advance how many distinct concepts are comprised by the test data. Furthermore, no feedback is given to the net as to which test data are instances of which concept.

Several issues are critical for the successful training of Kohonen nets. For example, the proper initialization of neurons is of great importance and has been the focus of many investigations. Another relevant aspect is the right choice of test data and measure of distance. The interested reader is referred to [3] and [4] for details.

Areas of Application

As with each innovative model of computation, the question arises about what kind of problems are suitable for being tackled therewith. It sometimes seems that perhaps too many applications for neural network technology are currently being explored. Figure 10 might help to illustrate for which problem domains neural nets have the potential to create better solutions than obtained with other approaches.

Problems can normally quite easily be classified as to whether they cause major difficulties for humans or for computers. We are not very fast and accurate in:
- performing calculations with large numbers;
- storing and retrieving vast amounts of data;
- sorting and searching large volumes of data.
On the other hand, humans typically have no problems whatsoever with:
- reading and comprehending text;
- listening and talking;
- identifying objects within pictures.
Furthermore, some issues are equally difficult for both humans and computers to handle. Examples thereof include:

- diagnosing malfunctions in complex systems;
- planning the proper steps for solving tasks;
- configuring large systems restricted by certain constraints.
These categories of problems are amenable to treatment by algorithmic data processing, symbolic knowledge processing and neural networks, respectively. It remains to explain the intrinsic differences between the second and third of these approaches. Figures 11 and 12 depict some critical aspects. The standard way of building knowledge bases systems is shown in Figure 11.

The goal is to fill a knowledge base (KB) in such a way that an inferencing mechanism (IM) can draw conclusions based on knowledge and data stored in a working memory (WM). Most important is knowledge which must have been acquired from human experts. Thus, domain specific expertise must be made explicit and formally represented by symbolic expressions like production rules, frames or logical formulas. In contrast with this, neural nets are made operational by presenting them with sample problems from the universe of discourse, as shown in Figure 12. They might be able to extract the critical features hidden in these examples and more or less automatically acquire the skills to solve a large variety of similar problems.
This approach has been successfully attempted for many challenging domains. Chapter 11 of reference [1] gives an impressive overview.

Figure 12

**Future Perspectives**

There is no doubt that neural network technology is here to stay. However, it will neither replace symbolic knowledge processing nor algorithmic data processing. These approaches will coexist and supplement each other. There are many opportunities for bringing these techniques together. To mention just one, consider test expressions as commonly found in programs. They typically look like:

- if \( I \leq 1000 \) then ....
- while KEY <> LIST(I) do ....

It is not hard to predict that future software systems might include constructs such as:

- if PLAN_17 applicable do ....
- while SOUND_OF_ENGINE ok do ....

Why not activate an expert system to check the applicability of PLAN_17, or trigger a neural net to monitor the sound waves created by an engine? Future generations of software systems will increasingly become hybrid compounds. The more different computing paradigms they can draw upon the better.

Neural networks have not yet reached a state of maturity that comes close to software or knowledge engineering. Still, the development of a neural net to solve practically relevant problem is more art than science. It requires a lot of experience, normally acquired through trial and error, to find:

- the right net architecture;
- the best suited training paradigm;
- the proper set of test data;
- the right tuning of the training pace;

and many other things. In sum, sound approaches for neural network engineering have yet to be developed and validated.

Currently, most neural nets are implemented in conventional programming languages and run on standard computer architecture. The power of typical workstations or even PCs is normally sufficient to do the job. On the other hand, there are less mundane computer architectures available, such as transputer nets or connectionistic machines. Their potential has not yet been extensively exploited and assessed in the development of neural networks. Therefore, what the best possible hardware platform for neural nets looks like is an open question. It would not be surprising if in the future special boards were to be available, which could be plugged into PCs or workstations to boost the computations required by neural nets.

**References**

Notes for Contributors

The prime purpose of the journal is to publish original research papers in the fields of Computer Science and Information Systems, as well as shorter technical research papers. However, non-refereed review and exploratory articles of interest to the journal’s readers will be considered for publication under sections marked as Communications or Viewpoints. While English is the preferred language of the journal papers in Afrikaans will also be accepted. Typed manuscripts for review should be submitted in triplicate to the editor.

Form of Manuscript

Manuscripts for review should be prepared according to the following guidelines.

- Use double-space typing on one side only of A4 paper, and provide wide margins.
- The first page should include:
  - title (as brief as possible);
  - author’s initials and surname;
  - author’s affiliation and address;
  - an abstract of less than 200 words;
  - an appropriate keyword list;
  - a list of relevant Computing Review Categories.
- Tables and figures should be on separate sheets of A4 paper, and should be numbered and titled. Figures should be submitted as original line drawings, and not photocopies.
- Mathematical and other symbols may be either handwritten or typed. Greek letters and unusual symbols should be identified in the margin, if they are not clear in the text.
- References should be listed at the end of the text in alphabetic order of the (first) author’s surname, and should be cited in the text in square brackets. References should thus take the following form:

Manuscripts accepted for publication should comply with the above guidelines, and may be provided in one of the following formats:

- in a typed form (i.e. suitable for scanning);
- as an ASCII file on diskette; or
- as a WordPerfect, \TeX or \LaTeX file; or
- in camera-ready format.

A page specification is available on request from the editor, for authors wishing to provide camera-ready copies. A styles file is available from the editor for Wordperfect, \TeX or \LaTeX documents.

Charges

Charges per final page will be levied on papers accepted for publication. They will be scaled to reflect scanning, typesetting, reproduction and other costs. Currently, the minimum rate is R20-00 per final page for camera-ready contributions and the maximum is R100-00 per page for contributions in typed format.

These charges may be waived upon request of the author and at the discretion of the editor.

Proofs

Proofs of accepted papers will be sent to the author to ensure that typesetting is correct, and not for addition of new material or major amendments to the text. Corrected proofs should be returned to the production editor within three days.

Note that, in the case of camera-ready submissions, it is the author's responsibility to ensure that such submissions are error-free. However, the editor may recommend minor typesetting changes to be made before publication.

Letters and Communications

Letters to the editor are welcomed. They should be signed, and should be limited to about 500 words.

Announcements and communications of interest to the readership will be considered for publication in a separate section of the journal. Communications may also reflect minor research contributions. However, such communications will not be refereed and will not be deemed as fully-fledged publications for state subsidy purposes.

Book reviews

Contributions in this regard will be welcomed. Views and opinions expressed in such reviews should, however, be regarded as those of the reviewer alone.

Advertisement

Placement of advertisements at R1000-00 per full page per issue and R500-00 per half page per issue will be considered. These charges exclude specialized production costs which will be borne by the advertiser. Enquiries should be directed to the editor.
Contents

GUEST CONTRIBUTION

Why all the Fuss About Neural Networks?
G Barth ................................................. 4

RESEARCH ARTICLES

The Placement of Subprograms by an Automatic Programming System
J P du Plessis and H J Messerschmidt ................................................. 9

An Investigation into the Separation of the Application from its User Interface
J H Greyling and P R Warren .................................................. 16

The Physical Correlates of Local Minima
L F A Wessels, E Barnard and E van Rooyen ................................................. 22

An Efficient Primal Simplex Implementation for the Continuous 2-Matching Problem
T H C Smith, T W S Meyer and L Leenen ................................................. 28

Concept Network Framework for a Multi-paradigm Knowledge Base
J Kambanis ................................................. 32

PEST - A Microcomputer Pascal Based Expert System Shell
A G Sartori-Angus and R Neville ................................................. 39

A Linda Solution to the Evolving Philosophers Problem
S E Hazelhurst ................................................. 44

TECHNICAL NOTE

Knowledge Representation using Formal Grammars
S H von Solms, E M Ehlers and D J Enslin ................................................. 54

COMMUNICATIONS AND REPORTS

Book Review & Books Received ................................................. 57
Cost-effective Visual Simulation Based on Graphics Workstations
A Caduri and D G Kourie ................................................. 58
A Method of Controlling Quality of Application Software
T D Crossman ................................................. 70
An Update on UNINET-ZA: The Southern African Academic and Research Network
V Shaw ................................................. 75