ELSEVIER

Information & Management 27 (1994) 303–313

INFORMATION & MANAGEMENT

Applications

Artificial neural networks and their business applications

Eldon Y. Li

Institute of Information Management, National Chung Cheng University, 160, San-Hsing, Ming-Hsiung, Chia-Yi 621, Taiwan, R.O.C.

Abstract

Artificial neural networks are increasingly popular in today's business fields. They have been hailed as the greatest technological advance since the invention of transistors. The purpose of this paper is to answer two of the most frequently asked questions: "What are neural networks?" "Why are they so popular in today's business fields?" The paper reviews the common characteristics of neural networks and discusses the feasibility of neural-net applications in business fields. It then presents four actual application cases and identifies the limitations of the current neural-net technology.

Keywords: Artificial neural networks; Network structures; Parallel processing; Learning methods; Business applications; Neural-net limitations

1. Introduction

Recently, applications of artificial neural networks have been increasing in business. More and more development tools have emerged on the market. Many neural-net systems have been shown to work well in identifying intricate patterns, learning from experience, reaching some conclusions, and making predictions. According to J. Clarke Smith, executive vice president of Sears Mortgage Corporation, neural-net systems have already been at work for over 10 years in the finance world. Today, they are widely applicable to risk management and forecasting [24]. Since the various neural-net systems now in use are implemented with mathematically sound principles, they hold out promise for future applications.

1.1. What is an artificial neural network?

An artificial neural network (ANN) does not emulate the thought processes and if/then logic of the human brain as done by an expert system. It mimics certain aspects of the information processing and physical structure of the brain with a web of neural connections (see Figure 1). Therefore, some writers classified it as a "microscopic", "white-box" system and an expert system as a "macroscopic", "black-box" system. An ANN consists of a large number of simple processing elements that are interconnected and layered. The biological neuron looks like a tree, except that between the trunk and the branches there is a large polygon shape which is the body of the cell, called the "soma". The soma is enclosed by a cell wall called "membrane". The tree branches

SSDI 0378-7206(94)00011-7

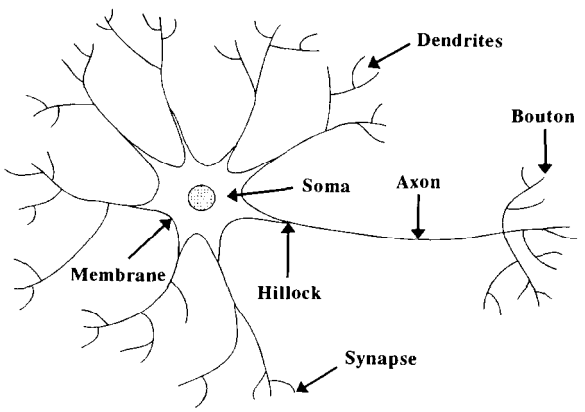

Fig. 1. The physical structure of a biological neuron.

are the "dendrites", which form a star shape and the tip of the branches are the "synapses". The tree trunk is the "axon" and the point of connection of the axon to the soma is the "hillock". Finally, the tree trunk extends to the root branches which are the "boutons". Synapses are areas of electrochemical contact between neurons. Through the synapses, the dendrites receive signals from other cells and transmit them to the soma. The soma adds up the incoming signals over time and at some level will respond to the inputs. When a neuron fires, the impulses are generated at the hillock, pass down the axon, and

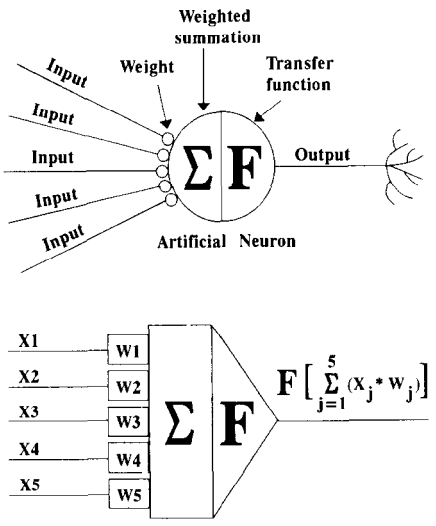

Fig. 2. The functions of an artificial neuron.

reach the boutons, then are sent on to the other neurons [25].

Similar to its biological counterpart, an ANN (see Figure 2) has each processing element (the neuron) receiving inputs from the other elements, the inputs are weighted and added, the result is then transformed (by a transfer function) into the output. The transfer function may be a step, sigmoid, or hyperbolic tangent function, among others.

In effect, ANNs are primitive learning devices. Their implementation may be in the form of hardware or software. In either case, the underlying concept is to assemble many single simple processors which interact through a dense web of interconnections. This network architecture, also known as "connectionism" [1], is unlike the conventional architecture of computer systems.

2. Characteristics of artificial neural networks

Conventionally, a computer operates through sequential linear processing technologies. They apply formulas, decision rules, and algorithms instructed by users to produce outputs from the inputs. Conventional computers are good at numerical computation. But ANNs improve their own rules; the more decisions they make, the better the decisions may become.

There are six main characteristics of ANN technology: the network structures, the parallel processing ability, the distributed memory, the fault tolerance ability, the collective solution, and the learning ability.

(1) Network structures: An ANN may have either a recurrent or nonrecurrent structure. A recurrent network [9,10] is a feedback network (see Figure 3a) in which the network calculates its outputs based on the inputs and feeds them back to modify the inputs. For a stable recurrent network, this process normally produces smaller and smaller output changes until the output become constant. If this process would not end, the network is unstable and is known as a chaotic system [2,7] – a system in which many Wall Street experts believe it can predict stock prices [20,28]. To create a stable network, the weight matrix

must be symmetrical with zeros on its main diagonal [5]. Moreover, the outputs may be fed back to middle layers to adjust the weights [8], similar to unsupervised learning. As for the nonrecurrent networks (see Figure 3b), data will flow in one direction, from input layer to output layer without any feedback loop: they are also called feed-forward networks. This type of networks has accounted for most existing ANN applications.

(2) Parallel processing ability: Each neuron in the ANN is a processing element similar to a Boolean logical unit in a conventional computer chip, except that a neuron's function is programmable. Computations required to simulate ANNs are mainly matrix ones, and the parallel structure of the interconnection between neurons facilitates such calculations. Figure 4 shows the calculations corresponding to each layer of a three-layer, one-middle-layer ANN. For simplicity, a single input vector is used. Each element in the vector is equivalent to one data point of the input variables which in reality should have multiple data points. Such matrix calculations require rapid computation, possible with neural-net chips now commercially available from Intel, Neural Semiconductor, and Bell Laboratories.

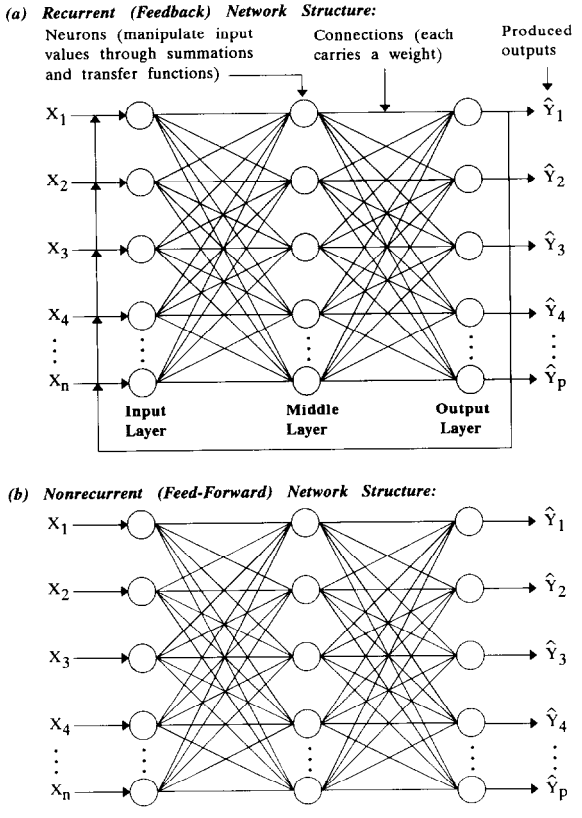

Fig. 3. Recurrent and nonrecurrent ANN structures.

(3) Distributed memory: The network does not store information in a central memory. Information is stored as patterns throughout the network structure. The state of neurons represents a short-term memory as it may change with the next input vector. The values in the weight matrix (the connections) form a long-term memory and are changeable only on a longer time basis [27]. Gradually, short-term memory will move into long-term memory and modify the network as a function of the input experience.

(4) Fault tolerance ability: The network's parallel processing ability and distributed memory make it relatively fault tolerant. In a neural computer, the failure of one or more parts may degrade the accuracy but it does not break the system. A system failure occurs only when all parts fail at the same time. This provides a measure of damage control.

(5) Collective solution: A conventional computer processes programmed instructions sequentially and one at a time. If a program is stopped in the middle of its execution, one can obtain a sensible answer which reflects exactly the computations that have been done so far. However, such a partial solution is meaningless with an ANN computer; it relies on the collective outputs of all the connected neurons. If the solution process is stopped before it is completed, the "partial answer" is probably nonsense to the user.

(6) Learning (or training) ability: An ANN, especially the nonrecurrent feed-forward one, is capable of applying learning rules to develop models of processes, while adapting the network to the changing environment and discovering useful knowledge implicit in received responses and/or stimuli. There are three possible learning methods: supervised, unsupervised, and reinforcement learning. In the first, the desired output for a set of training inputs is provided to the network; thus it learns by example. This is used

to train a network for a specific task. Unsupervised learning is conducted when there is no evaluation of performance provided to the network. Reinforcement learning is a hybrid method, the network is given a scalar evaluation signal instead of being told the desired output, and evaluations can be made intermittently instead of with every training input.

3. Examples of learning methods

Most business applications of ANN today use supervised learning method. To train a network, there must be a training set (containing inputs and target outputs) and a learning rule. Among the many rules, the most popular is the backpropagation algorithm [21] which is based on partial

Input Layer: ***(with n nodes, each has no weight and no transfer function:)***

$$\text{Input Vector} = \underset{\sim}{X} = \text{Output Vector} = [\, x_1 \;\; x_2 \;\; \cdots \;\; x_n \,]$$

Middle Layer: ***(with m nodes, each has n weights and one trasnfer function:)***

$$\text{Weight Matrix} = \underset{\sim}{W} = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1m} \\ w_{21} & w_{22} & \cdots & w_{2m} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ w_{n1} & w_{n2} & \cdots & w_{nm} \end{bmatrix}$$

$$\text{Transfer Functions} = \mathbf{F} = (f_1, f_2, \ldots, f_m)$$

$$\text{Output Matrix} = \mathbf{F}[\underset{\sim}{X}\ \underset{\sim}{W}]$$

$$= \left[f_1\left(\sum_{i=1}^{n} x_i\, w_{i1}\right), f_2\left(\sum_{i=1}^{n} x_i\, w_{i2}\right), \ldots, f_m\left(\sum_{i=1}^{n} x_i\, w_{im}\right) \right]$$

Output Layer: ***(with p nodes, each has m weights and one transfer function:)***

$$\text{Weight Matrix} = \underset{\sim}{V} = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1p} \\ v_{21} & v_{22} & \cdots & v_{2p} \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ v_{m1} & v_{m2} & \cdots & v_{mp} \end{bmatrix}$$

$$\text{Transfer Functions} = \mathbf{T} = (t_1, t_2, \ldots, t_p)$$

$$\text{Output Vector} = \underset{\sim}{\hat{Y}} = [\, \hat{y}_1 \;\; \hat{y}_2 \;\; \cdots \;\; \hat{y}_p \,] = \mathbf{T}\left[\mathbf{F}[\underset{\sim}{X}\ \underset{\sim}{W}]\,\underset{\sim}{V}\right]$$

$$= \left[t_1\left\{\sum_{j=1}^{m} \left[f_j\left(\sum_{i=1}^{n} x_i\, w_{ij}\right)\right] v_{j1}\right\},\right.$$

$$t_2\left\{\sum_{j=1}^{m} \left[f_j\left(\sum_{i=1}^{n} x_i\, w_{ij}\right)\right] v_{j2}\right\}, \ldots\ldots$$

$$\left.\ldots\ldots, t_p\left\{\sum_{j=1}^{m} \left[f_j\left(\sum_{i=1}^{n} x_i\, w_{ij}\right)\right] v_{jp}\right\}\right]$$

Fig. 4. Matrix calculations in a one-middle-layer neural network.

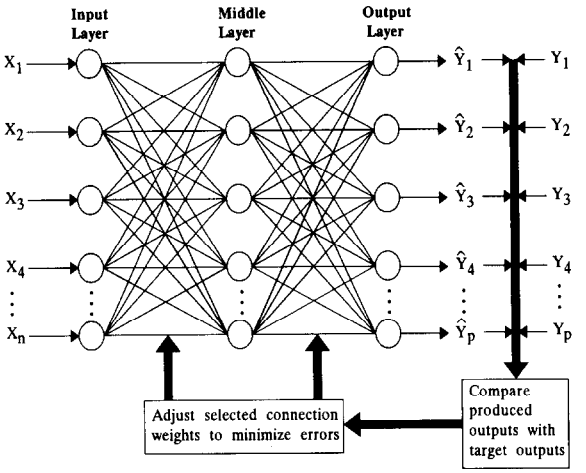

Fig. 5. Learning in a feed-forward neural network.

derivatives with momentum to approximate the direction of weight change that gives the most improvement in the output error. The inputs and outputs can be either discrete or continuous. Each input vector in the training set is matched with its desired output vector to form a training pair. During the learning process (see Figure 5), the training pairs are fed into the network one by one. The initial values of the connection weights are set randomly to small numbers. The network derives its responses and compares them with the desired ones. If there is an error, the system adjusts (increases or decreases) the weights by a small amount in the direction identified by the predefined learning rule. These adjustments are continued until the error begins to increase or is reduced to an acceptable level. Then, the weights are frozen alone and the training is completed. In effect, the supervised network is equivalent to a multivariate system of equations that maps input data to required output data.

In a situation where training pairs are difficult, if not impossible, to obtain, unsupervised learning [13] is necessary. The training set for this process consists solely of input vectors. The training algorithm modifies the connection weights to produce output vectors that are consistent. The technique extracts the statistical properties of the training vectors and groups them into classes in which each produces the same pattern of output. An unsupervised network can be used to train robotic arm movement. In this application, the network can develop patterns of transformations to orient the robotic arm through a training session conducted by a human operator. The equations of such complex inverse transformations do not need to be derived or programmed. Natural or unforeseen changes in the robot motion will automatically be incorporated in the network, allowing robots to learn to move in less precisely mapped environments – a task far too complex to be programmed algorithmically for general cases [12].

Another more complex application of unsupervised learning was reported by McDonnell Douglas in training aircraft's automatic flight control system. The company let top pilots fly a F-15 Eagle plane with simulated damage. The neural-net based flight controller learned to reconfigure the aircraft by taking notes on how top pilots react to emergency situations and how the aircraft responds. As the pilot completed the mission and landed the disabled aircraft, the network use its notes to come up with a model of its own. Next time around, the network would be able to compensate for the damages, letting the pilot guide the damaged plane as if nothing happened [3].

4. Feasibility of business applications

ANNs can be applied to many problems that are solved conventionally by statistical and management science techniques. In fact, the common characteristics enable ANNs to solve these problems better and faster than conventional techniques, even without human intervention. Moreover, they make it possible to model very complex decision tasks so easily and simply that little theoretical knowledge is required of the ANN users.

ANN tasks can be classified into the following categories:

(1) Approximation: To determine the weights that minimize the (least-square or absolute) error distance between the produced output and the target output [15]. This is somewhat equivalent to regression analysis in statistics, using an analytical

procedure to solve the normal equations and to find the regression coefficients (the equivalent to the connection weights of a trained network).

(2) Optimization: To determine the optimal solution to an NP-complete (nondeterministic polynomial) problem, such as the travelling salesperson problem [11]. This is equivalent to linear and integer programming in management sciences, where the objective functions are optimized using a heuristic search procedure.

(3) Classification: To classify an object characterized by its input vector into one of different categories or groups. The input vector may have continuous or discrete values. This is similar to discriminant analysis in nonparametric multivariate statistics.

(4) Prediction: To predict the output values from the input values. While the input values may be continuous or discrete, the output values are continuous; this makes it different from a classification task, being equivalent to making predictions and forecasts in multivariate statistics. However, the characteristics of an ANN allow it to represent a prediction or forecasting model, such a process is far too complex for a human decision maker.

(5) Generalization: To analyze the association between and within input attributes to extract statistical properties of the training set and to develop generalized patterns into which the objects are classified. Once the patterns are developed using error-free input data, noisy input patterns can be recognized and corrected; this makes the outputs of the classification and prediction tasks much more accurate than those produced by conventional techniques.

(6) Relation: To analyze how the input data are clustered into different groups and the relationships between and within input attributes in each group. This is comparable to factor analysis and cluster analysis in statistics, except that nonlinear relationships are allowed.

(7) Abstraction: To filter noise out of imperfect inputs, thereby increasing its integrity. This is somewhat similar to exploratory data analysis; outliers and items not significantly related to the target outputs are identified and removed.

(8) Adaptiveness: To adjust the connection weights in the network automatically as soon as a new training vector is fed into the network. This makes the network adaptive to an ever-changing dynamic environment. Conventionally, additional effort must be devoted to make such adaptive processes happen automatically. Human intervention in still often required.

Given the above capabilities, there is no doubt that ANNs are feasible for business applications. Many phenomena that are difficult to describe can be modeled by ANNs, if carefully designed.

5. Examples of business applications

There are many applications of ANNs in today's business. Financial institutions are improving their decision making by enhancing the interpretation of behavioral scoring systems and developing superior ANN models of credit card risk and bankruptcy [14,22]. Securities and trading houses are developing and improving their forecasting techniques and trading strategies with ANNs. Insurance companies are managing risk better by using ANNs to develop a model of top underwriters and using this as a training and evaluation tool for other underwriters. Manufacturers are improving their product quality through predictive process control systems using ANNs [18]. Oil and gas corporations are learning more from their data by using ANNs to interpret seismic signals and sub-surface images to improve their exploration effort. Four actual ANN applications are now described.

5.1. Airline security control

With the increasing threat of terrorism, airline passengers' bags in international airports such as New York, Miami, and London go through an unusually rigorous inspection before being loaded into the cargo bay [4]. In addition to using metal detector and x-ray station to detect metal weapons, these airports use ANNs to screen for plastic explosives. They use a detection system which bombards the luggage with neutrons and monitors the gamma rays that are emitted in response. The network then analyzes the signal to decide

whether the response predicts an explosive. The purpose of this operation is to detect explosives with a 95 percent probability, while minimizing the number of false alarms.

Detecting explosive using gamma rays is not simple since different chemical elements release different frequencies. Explosive materials are rich in nitrogen, but so are some benign substances, including protein-rich materials, such as wool and leather. Though an abundance of gamma rays at nitrogen's frequency raises some suspicion, it is difficult to make a distinction. To minimize the classification error, supervised training was conducted. The ANN was fed with a batch of instrument reading as well as the information on whether explosives were indeed present. The trained network were able to achieve its intended purpose. The entire security system can handle 600 to 700 bags per hour and the network raises false alarms on only 2 percent of the harmless bags at the 95 percent detection point. This reduction in false alarms translates into many less bags that must be opened and examined each day. In turn, it reduces the cost of airport operations, increases the efficiency of the check-in process, and improves the satisfaction of customers.

5.2. *Investment management and risk control*

Neural Systems Inc. [17] makes use of a supervised network to mimic the recommendations of money managers on the optimal allocation of assets among Treasury instruments. The application demonstrated how well an ANN can be trained to recognize the shape and evolution of the interest-yield curve and to make recommendations as to long or short positions in the US Treasury market.

The network was trained on measured and calculated economic indicators, such as the evolution of interest rates, price changes, and the shape and speed of the change of the yields curves. The network could then determine the optimal allocation among segments in various Treasury instruments being measured against a benchmark or comparator performance index. It could determine also the dynamic relationship between different variables in portfolio management and risk control. Consequently, it allowed more active control of portfolio's level of certainty. Based on the experience gained with this application, another ANN with a higher level of complexity was subsequently developed.

5.3. *Prediction of thrift failures*

Professor Linda M. Salchenberger and her colleagues at the Loyola University of Chicago have developed an ANN to predict the financial health of savings and loan associations [22]. They identified many possible inputs to the network. Through stepwise regression analyses, 5 significant variables were identified (out of 29). These variables were the ratios of: net worth/total assets, repossessed assets/total assets, net income/gross income, net income/total assets, cash plus securities/total assets. They ratios were selected to measure, respectively, capital adequacy, asset quality, management efficiency, earnings, and liquidity.

After identifying the input variables, they conducted some experiments and selected a single middle layer, feed-forward, backpropagation network consisting of 5 input nodes, 3 middle layer nodes, and one output node (see Figure 6). The output node was interpreted as the probability that an institution was classified as failed or surviving.

To train the network, supervised learning was conducted with training sets consisting of the five financial ratios and the corresponding failed or

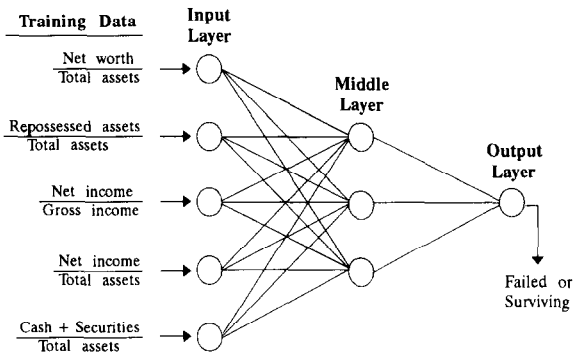

Fig. 6. Neural network for predicting thrift failures.

surviving result from 100 failures and 100 surviving S and L institutions between January, 1986 to December, 1987. The result showed the three-layer ANN gained more predictive power over logit model. The latter is equivalent to a two-layer (no middle-layer) network.

5.4. Prediction of stock price index

With limited knowledge about the stock market and with only data available from a public library, Ward Systems Group, Inc. [26] created an example showing how one might set up an ANN application to predict stock market behavior. The first step was to decide what to predict or classify (i.e., the target outputs). Obviously there are many possible outputs that could be predicted, such as turning points, market direction, etc. For this application, the next month's average Standard and Poor's stock price index was selected.

The next step was to consider which input facts or parameters are necessary or useful for predicting the target outputs. In this case, the stock price index for the current month was chosen because it should be an important factor in predicting next month's index. In addition, nine other publicly available economic indicators were selected: unadjusted retail sales, average three month Treasury bill rate, total U.S. Government securities, industrial production index, New York gold price, outstanding commercial paper and acceptances, Swiss Franc value, U.S. Government receipts, and U.S. Government expenditures (see Figure 7).

Next, the case characteristics for the problem were entered into the system. These included the defining characteristics (the names of the input parameters) and the classifying characteristics (the names of the output results). Finally, examples of previous results were entered in order to train the network. These case histories contain information for all the months in the years of 1974 to 1979. The goal is to see if the system could predict the monthly stock price indexes in 1980.

After several hours of training, the network was able to predict the next month's stock price index for all of 1980. The result has shown that

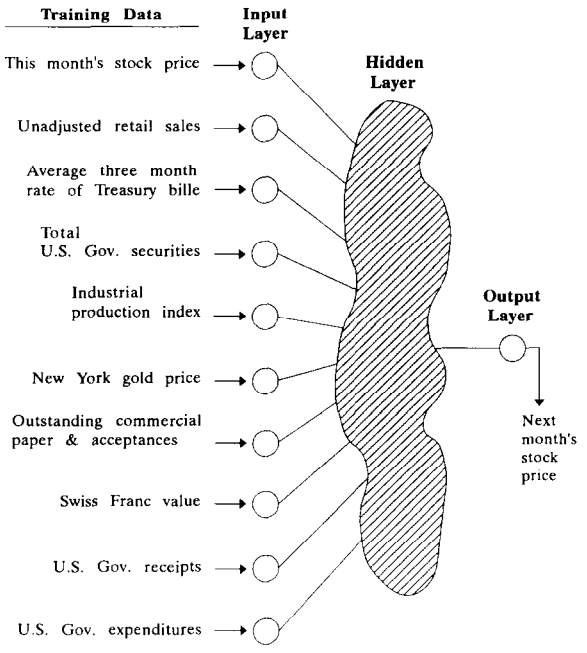

Fig. 7. A neural network for predicting stock price.

such neural system can produce the first 8 month predictions with less than 3.2% average absolute error and the entire 12 month predictions with only 4% average error. Therefore, through a carefully designed ANN, it is possible to predict the volatile stock market.

6. Limitations of artificial neural networks

Artificial neural network is undoubtedly a powerful tool for decision making. But there are several weaknesses in its use.

(1) ANN is not a general-purpose problem solver. It is good at complex numerical computation for the purposes of solving system of linear or non-linear equations, organizing data into equivalent classes, and adapting the solution model to environmental changes. However, it is not good at such mundane tasks as calculating payroll, balancing checks, and generating invoices. Neither is it good at logical inference – a job suited for expert systems. Therefore, users must know when a problem could be solved with an ANN.

(2) There is no structured methodology available for choosing, developing, training, and verifying an ANN [23]. The solution quality of an ANN is known to be affected by the number of layers, the number of neurons at each layer, the transfer function of each neuron, and the size of the training set. One would think that the more data in the training set, the better the accuracy of the output. But, this is not so. While too small a training set will prohibit the network from developing generalized patterns of the inputs, too large a one will break down the generalized patterns and make the network sensitive to input noise. In any case, the selection of these parameters is more of an art than a science. Users of ANNs must conduct experiments (or sensitivity analyses) to identify the best possible configuration of the network. This calls for easy-to-use and easy-to-modify ANN development tools that are gradually appearing on the market.

(3) There is no single standardized paradigm for ANN development. Because of its interdisciplinary nature, there have been duplicating efforts spent on ANN research. For example, the backpropagation learning algorithm was independently developed by three groups of researchers in different times: Werbos [29], Parker [19], and Rumelhart, Hinton, and Williams [21]. To resolve this problem, the ANN community should establish a repository of available paradigms to facilitate knowledge transfer between researchers.

Moreover, to make an ANN work, it must be tailored specifically to the problem it is intended to solve. To do so, users of ANN must select a particular paradigm as the starting prototype. However, there are many possible paradigms. Without a proper training, users may easily get lost in this. Fortunatcly, most of the ANN development tools commercially available today provide scores of sample paradigms that work on various classes of problems. A user may follow the advice and tailor it to his or her own needs.

(4) The output quality of an ANN may be unpredictable regardless of how well it was designed and implemented. This may not be the case for finding the solution to a problem with linear constraints in which the solution, if found, is guaranteed to be the global optimum. However, many problems have a non-linear region of feasible solutions. A solution to a non-linear problem reached by the ANN may not be the global optimum. Moreover, there is no way to verify that an ANN is correct unless every possible input is tried: such exhaustive testing is impractical, if not impossible. In a mission-critical application, one should develop ANN solutions in parallel with the conventional ones for direct comparison. Both types of systems should be run for a period of time, long enough to make sure that the ANN systems are error-free before they are used in real situations.

(5) Most ANN systems are not able to explain how they solve problems. The current ANN implementations are based primarily on random collectivity between processing elements (the individual "neurons"). As a result, the user may be able to verify a network's output but not to trace a system's flow of control [16]. Recently, S.I. Gallant [6] demonstrated that an explanation ability can be incorporated into an ANN. Further development of this is bound to attract more prospective users into the ANN bandwagon.

7. Conclusion

The field of ANN went through a dormant period during the 1970's, because the early single-layer models were fundamentally flawed. Soon after, some multi-layer and trainable ANN models emerged in the early 1980's. Despite having some inherent limitations, ANNs have been increasingly popular since then. They are feasible for those business applications which require the solution of very complex system of equations, recognizing patterns from imperfect inputs, and adapting decisions to changing environment.

Philip D. Wasserman of ANZA Research, Inc. envisions "artificial neural networks taking their place alongside of conventional computation as an adjunct of equal size and importance" [27]. Indeed, digital computers will always be needed to compute payrolls, manage inventory, and schedule production. As ANN software packages

become increasingly user-friendly, they will attract more and more novice users.

References

[1] Aeh, R.K. "Connectionism", *Journal of System Management*, April 1989, p. 23.

[2] Babloyantz, A. and Destexhe, A. "Chaos in Neural Networks", In M. Caudill and C. Butler (Eds.) *IEEE First International Conference on Neural Networks* (Vol. 4). Piscataway, NJ: IEEE Service Center, pp. 31–39.

[3] Berardinis, L.A. "Untangling the Mystery of Neural Networks", *Machine Design*, Vol. 65, No. 13, June 25, 1992, pp. 55–59.

[4] Brody, H. "The Neural Computer", *Technology Review*, August/September 1990, pp. 43–49.

[5] Cohen, M.A. and Grossberg, S.G. "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, 1983, pp. 815–826.

[6] Gallant, S.I. "Connectionist Expert Systems", *Communications of the ACM*, Vol. 31, No. 2, February 1988, pp. 152–169.

[7] Harth, E. "Order and Chaos in Neural Systems: An Approach to the Dynamics of Higher Brain Functions", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-13, September 1983, pp. 782–789.

[8] Hinton, G.E. and Sejnowski, T.J. "Learning and Relearning in Boltzmann Machines", In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing* (Vol. 1). Cambridge, MA: The MIT Press, 1986, pp. 282–317.

[9] Hopfield, J.J. "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of the National Academy of Science*, Vol. 79, 1982, pp. 2554–2558.

[10] Hopfield, J.J. "Neurons with Graded Response Have Collective Computational Properties Like Those of Two-State Neurons", *Proceedings of the National Academy of Science*, Vol. 81, 1984, pp. 3088–3092.

[11] Hopfield, J.J. and Tank, D. "Neural Computation of Decisions in Optimization Problems", *Biological Cybernetics*, Vol. 52, 1985, pp. 147–152.

[12] Josin, G. "Integrating Neural Networks with Robots", *AI Expert*, August 1988, pp. 50–54.

[13] Kohonen, T., Self-Organization and Associative Memory. *Series in Information Science*, Vol. 8, Berlin: Springer-Verlag, 1984.

[14] Koster, A., Sondak, N.E., and Bourbia, W. "A Business Application of Artificial Neural Network Systems", *Journal of Computer Information Systems*, Vol. 32, No. 2, Winter 1991, pp. 3–9.

[15] Kung, S.Y., Diamantaras, K., Mao, W.D. and Taur, J.S. "Generalized Perceptron Networks with Nonlinear Discriminant Functions", In R.J. Mammone and Y. Zeevi (Eds.), *Neural Networks: Theory and Applications*. Boston: Academic Press, Inc., 1991, pp. 245–279.

[16] Liebowitz, J. and De Salvo, D. *Structuring Expert Systems*. Englewood Cliffs, NJ: Prentice-Hall, 1989.

[17] Neural Systems Inc. *Neural Systems Journal for Genesis Users*, Vol. 2, No. 1, Spring/Summer 1990. (Available from Neural Systems Inc., 2827 West 43rd Ave., Vancouver, British Columbia, CANADA V6N 3H9, Tel: 604–263-3667.)

[18] NeuralWare, Inc. *Pushing the Frontiers of Neural Computing*. Pittsburgh, PA: NeuralWare, Inc., 1991. (Available from NeuralWare, Inc., Penn Center West, Building IV, Suite 227, Pittsburgh, PA 15276, Tel: 412–787-8222.)

[19] Parker, D.B. Learning Logic. *Invention Report #S81-46*, File 1, Office of Technology Licensing. Stanford, CA: Stanford University, 1982.

[20] Peters, E.E. *Chaos and Order in the Capital Markets*. New York: John Wiley, 1991.

[21] Rumelhart, D.E., Hinton, G.E. and Williams, R.J. "Learning Internal Representation by Error Propagation", In D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing* (Vol. 1). Cambridge, MA: The MIT Press, 1986, pp. 318–362.

[22] Salchenberger, L.M., Cinar, E.M. and Lash, N.A. "Neural Networks: A New Tool for Predicting Thrift Failures", *Decision Sciences*, Vol. 23, No. 4, July/August 1992, pp. 899–916.

[23] Shriver, B. "Artificial Neural Systems", *IEEE Computer*, Vol. 12, No. 3, March 1988, pp. 8–9.

[24] Smith, J.C. "A Neural Network – Could It Work for You?" *Financial Executive*, Vol. 6, No. 3, May/June 1990, pp. 26–30.

[25] Stanley, J. *Introduction to Neural Networks*. Sierra Madre, CA: California Scientific Software, 1989.

[26] Ward Systems Group, Inc. *101 Ways to Use NeuroShell*. Frederick, MD: Ward Systems Group, Inc., 1990. (Available from Ward Systems Group, Inc., 245 W. Patrick Street, Frederick, MD 21701, Tel: 301–662-7950.)

[27] Wasserman, P.D. *Neural Computing: Theory and Practice*. New York: Van Nostrand Reinhold, 1989.

[28] Weiss, G. "Chaos Hits Wall Street – The Theory, That Is", *Business Week*, November 2, 1992, pp. 138–140.

[29] Werbos, P.J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph.D. Thesis. Cambridge, MA: Harvard University, 1974.

Eldon Y. Li is Professor and Founding Director of Institute of Information Management at National Chung Cheng University in Taiwan. He is formerly a Professor and Coordinator of MIS program at School of Business, California Polytechnic State University, San Luis Obispo. He holds a bachelor degree from National Chengchi University in Taiwan and

M.S. and Ph.D. degrees from Texas Tech University. He have received two "Best Paper" awards, one from the *Quality Data Processing* journal in 1990 and the another from the *ACME Proceedings* in 1991. He has provided consulting services to many firms for a variety of software projects and served as a management consultant to the clientele of the U.S. Small Business Administration. He is a former software quality specialist at Bechtel Corporation Information Services Division and a former visiting software scientist at IBM Corporation. His current research interest lies in human factors in information technology (IT), strategic IT planning, software engineering, quality assurance, and information management. He is a Certified Data Educator (CDE) and is Certified in Production and Inventory Management (CPIM). Being a member of ACM, ACME, IACIS, DSI, NACISPA, and TIMS, he have published in *Information & Management, Information Resources Management Journal, Journal of Management Information Systems, Journal of Systems Management, Quality Data Processing, The Journal of Computer Information Systems, Group & Organization Studies, Public Personnel Management*, and *Simulation & Gaming*. He currently serves as a member of the editorial board for *The Journal of Quality Assurance Institute*.