ONLINE CHAPTER

6 Neural Networks for Data Mining

Learning Objectives

- Understand the concept and different types of artificial neural networks (ANN)
- Learn the advantages and limitations of ANN
- Understand how backpropagation neural networks learn
- Understand the complete process of using neural networks
- Appreciate the wide variety of applications of neural networks

Neural networks have emerged as advanced data mining tools in cases where other techniques may not produce satisfactory predictive models. As the term implies, neural networks have a biologically inspired modeling capability, but are essentially statistical modeling tools. In this chapter, we study the basics of neural network modeling, some specific applications, and the process of implementing a neural network project.

6.1 OPENING VIGNETTE: USING NEURAL NETWORKS TO PREDICT BEER FLAVORS WITH CHEMICAL ANALYSIS

Coors Brewers Ltd., based in Burton-upon-Trent, Britain's brewing capital, is proud of having the United Kingdom's top beer brands, a 20 percent share of the market, years of experience, and of the best people in the business. Popular brands include Carling (the country's best-selling lager), Grolsch, Coors Fine Light Beer, Sol, and Korenwolf.

PROBLEM

Today's customer is confronted with variety of options regarding what he or she drinks. A drinker's choice depends on various factors, such as mood, venue, and occasion. The goal of Coors is to ensure that the customer chooses a Coors brand every time.

According to Coors, creativity is the key to being successful in the long term. To be the customer's choice brand, Coors needs to be creative and anticipative about the customer's ever-changing moods. An important issue with beers is the flavor; each beer has a distinctive flavor. These flavors are mostly determined through panel tests. However, such tests take time. If Coors could understand the beer flavor based solely on its chemical composition, it would open up new avenues to create beer that would suit customer expectations.

The relationship between chemical analysis and beer flavor is not clearly understood yet. Substantial data exists about its chemical composition and sensory analysis. Coors needed a mechanism to link those two together. Neural networks were applied to create the link between chemical composition and sensory analysis.

SOLUTION

Over the years, Coors Brewers Ltd. has accumulated a significant amount of data related to the final product analysis, which has been supplemented by sensory data provided by the trained in-house testing panel. Some of the analytical inputs and sensory outputs are shown here:

Analytical Data: Inputs	*Sensory Data: Outputs*
Alcohol	Alcohol
Color	Estery
Calculated bitterness	Malty
Ethyl acetate	Grainy
Iso butyl acetate	Burnt
Ethyl butyrate	Hoppy
Iso amyl acetate	Toffee
Ethyl hexanoate	Sweet

A single neural network, restricted to a single quality and flavor, was first used to model the relationship between the analytical and sensory data. The neural network was based on a package solution supplied by NeuroDimension, Inc. (**nd.com**). The neural network consisted of a multilayer perceptron (MLP) architecture with two hidden layers. Data were normalized within the network, thereby enabling comparison between the results for the various sensory outputs. The neural network was trained (to learn the relationship between the inputs and outputs) through the presentation of many combinations of relevant input/output combinations. When there was no observed improvement in the network error in the last 100 epochs, training was automatically terminated. Training was carried out 50 times to ensure that a considerable mean network error could be calculated for comparison purposes. Prior to each training run, a different training and cross-validation data set was presented by randomizing the source data records, thereby removing any bias.

This technique produced poor results, due to two major factors. First, concentrating on a single product's quality meant that the variation in the data was pretty low. The neural network could not extract useful relationships from the data. Second, it was probable that only one subset of the provided inputs would have an impact on the selected beer flavor. Performance of the neural network was affected by "noise" created by inputs that had no impact on flavor.

A more diverse product range was included in the training range to address the first factor. It was more challenging to identify the most important analytical inputs. This challenge was addressed by using a software switch that enabled the neural network to be trained on all possible combinations of inputs. The switch was not used to disable a significant input; if the significant input were disabled, we could expect the network error to increase. If the disabled input was insignificant, then the network error would either remain unchanged or be reduced due to the removal of noise. This approach is called an *exhaustive search* because all possible combinations are evaluated. The technique, although conceptually simple, was computationally impractical with the numerous inputs; the number of possible combinations was 16.7 million per flavor.

A more efficient method of searching for the relevant inputs was required. A genetic algorithm was the solution to the problem. A genetic algorithm was able to manipulate the different input switches in response to the error term from the neural network. The objective of the genetic algorithm was to minimize the network error term. When this minimum was reached, the switch settings would identify the analytical inputs that were most likely to predict the flavor.

RESULTS

After determining what inputs were relevant, it was possible to identify which flavors could be predicted more skillfully. The network was trained using the relevant inputs previously identified multiple times. Before each training run, the network data were randomized to ensure that a different training and cross-validation data set was used. Network error was recorded after each training run. The testing set used for assessing the performance of the trained network contained approximately 80 records out of the sample data. The neural network accurately predicted a few flavors by using the chemical inputs. For example, "burnt" flavor was predicted with a correlation coefficient of 0.87.

Today, a limited number of flavors are being predicted by using the analytical data. Sensory response is extremely complex, with many potential interactions and hugely variable sensitivity thresholds. Standard instrumental analysis tends to be of gross parameters, and for practical and economical reasons, many flavor-active compounds are simply not measured. The relationship of flavor and analysis can be effectively modeled only if a large number of flavor-contributory analytes are considered. What is more, in addition to the obvious flavor-active materials, mouth-feel and physical contributors should also be considered in the overall sensory profile.

With further development of the input parameters, the accuracy of the neural network models will improve.

Sources: C.I. Wilson and L. Threapleton, "Application of Artificial Intelligence for Predicting Beer Flavours from Chemical Analysis," *Proceedings of the 29th European Brewery Congress*, Dublin, May 17–22, 2003, **neurosolutions.com/resources/apps/beer.html** (accessed April 2006); R. Nischwitz, M. Goldsmith, M. Lees, P. Rogers, and L. MacLeod, "Developing Functional Malt Specifications for Improved Brewing Performance," *The Regional Institute Ltd.,* **regional.org.au/au/abts/1999/nischwitz.htm** (accessed April 2006); and **coorsbrewers.com** (accessed April 2006).

Questions for the Opening Vignette

1. Why is beer flavor important to the profitability of Coors?
2. What is the objective of the neural network used at Coors?
3. Why were the results of the Coors neural network initially poor, and what was done to improve the results?

4. What benefits might Coors derive if this project is successful?
5. What modifications would you provide to improve the results of beer flavor prediction?

WHAT WE CAN LEARN FROM THIS VIGNETTE

As you will see in this chapter, applications of neural networks abound in many areas, from standard business problems of assessing creditworthiness of individuals to manufacturing, security, and health applications. This vignette illustrates an innovative application in a setting where human expertise may be considered the only way to assess quality. The vignette shows that the imagination of an analyst is the only limitation to exploring applications of data mining techniques in general and neural networks in particular. We also learn that in many real-life applications, we have to combine more than one advanced technique in order to create a useful application. In this particular situation, neural networks were combined with genetic algorithms, but other combinations are possible as well.

6.2 BASIC CONCEPTS OF NEURAL NETWORKS

Neural networks represent a brain metaphor for information processing. These models are biologically inspired rather than an exact replica of how the brain actually functions. Neural networks have been shown to be very promising systems in many forecasting applications and business classification applications due to their ability to "learn" from the data, their nonparametric nature (i.e., no rigid assumptions), and their ability to generalize. **Neural computing** refers to a pattern recognition methodology for machine learning. The resulting model from neural computing is often called an **artificial neural network (ANN)** or a *neural network*. Neural networks have been used in many business applications for pattern recognition, forecasting, prediction, and classification. Neural network computing is a key component of any data mining (see Chapter 4) tool kit. Applications of neural networks abound in finance, marketing, manufacturing, operations, information systems, and so on. Therefore, we devote this chapter to developing a better understanding of neural network models, methods, and applications.

The human brain possesses bewildering capabilities for information processing and problem solving that modern computers cannot compete with in many aspects. It has been postulated that a model or a system that is enlightened and supported by the results from brain research, with a structure similar to that of biological neural networks, could exhibit similar intelligent functionality. Based on this bottom-up postulation, ANN (also known as connectionist models, parallel distributed processing models, neuromorphic systems, or simply neural networks) have been developed as biologically inspired and plausible models for various tasks.

Biological neural networks are composed of many massively interconnected primitive biological neurons. Each neuron possesses *axons* and *dendrites*, finger-like projections that enable the neuron to communicate with its neighboring *neurons* by transmitting and receiving electrical and chemical signals. More or less resembling the structure of their counterparts, ANN are composed of interconnected, simple processing elements called *artificial neurons*. In processing information, the processing elements in an ANN operate concurrently and collectively in a similar fashion to biological neurons. ANN possess some desirable traits similar to those of biological neural networks, such as the capabilities of learning, self-organization, and fault tolerance.

Coming along a winding journey, ANN have been investigated by researchers for more than half a century. The formal study of ANN began with the pioneering work of McCulloch and Pitts in 1943. Stimulated by results of biological experiments and observations, McCulloch and Pitts (1943) introduced a simple model of a binary artificial neuron that captures some functions of a living neuron. Considering information processing machines as a means for modeling the brain, McCulloch and Pitts built their neural networks model using a large number of interconnected binary artificial neurons. Led by a school of researchers, neural network research was quite popular in the late 1950s and early 1960s. After a thorough analysis of an early neural network model (called the **perceptron**, which used no hidden layer) as well as a pessimistic evaluation of the research potential by Minsky and Papert in 1969, the interest in neural networks diminished.

During the past two decades, there has been an exciting resurgence in the studies of ANN due to the introduction of new network topologies, new activation functions, and new learning algorithms, as well as progress in neuroscience and cognitive science. On the one hand, advances in theory and methodology have overcome many obstacles that hindered neural network research a few decades ago. Evidenced by the appealing results of numerous studies, neural networks are gaining acceptance and popularity. On the other hand, as complex problems solvers, ANN have been applied to solve numerous problems in a variety of application settings. The desirable features in neural information processing make neural networks attractive for solving complex problems. The initial success in neural network applications has inspired renewed interest from industry and business.

BIOLOGICAL AND ARTIFICIAL NEURAL NETWORKS

The human brain is composed of special cells called **neurons**. These cells do not die when a human is injured (all other cells reproduce to replace themselves and then die). This phenomenon may explain why we retain information. Information storage spans sets of neurons. The estimated number of neurons in a human brain is 50 to 150 billion, of which there are more than 100 different kinds. Neurons are partitioned into groups called *networks*. Each network contains several thousand highly interconnected neurons. Thus, the brain can be viewed as a collection of neural networks.

The ability to learn and react to changes in our environment requires intelligence. The brain and the central nervous system control thinking and intelligent behavior. People who suffer brain damage have difficulty learning and reacting to changing environments. Even so, undamaged parts of the brain can often compensate with new learning.

A portion of a network composed of two cells is shown in Figure 6.1. The cell itself includes a **nucleus** (the central processing portion of the neuron). To the left of cell 1, the **dendrites** provide input signals to the cell. To the right, the **axon** sends output signals to cell 2 via the axon terminals. These axon terminals merge with the dendrites of cell 2. Signals can be transmitted unchanged, or they can be altered by synapses. A **synapse** is able to increase or decrease the strength of the connection from neuron to neuron and cause excitation or inhibition of a subsequent neuron. This is where information is stored.

An ANN model emulates a biological neural network. Neural computing actually uses a very limited set of concepts from biological neural systems (see Technology Insights 6.1). It is more of an analogy to the human brain than an accurate model of it. Neural concepts are usually implemented as software simulations of the massively parallel processes that involve processing elements (also called *artificial neurons*, or *neurodes*)

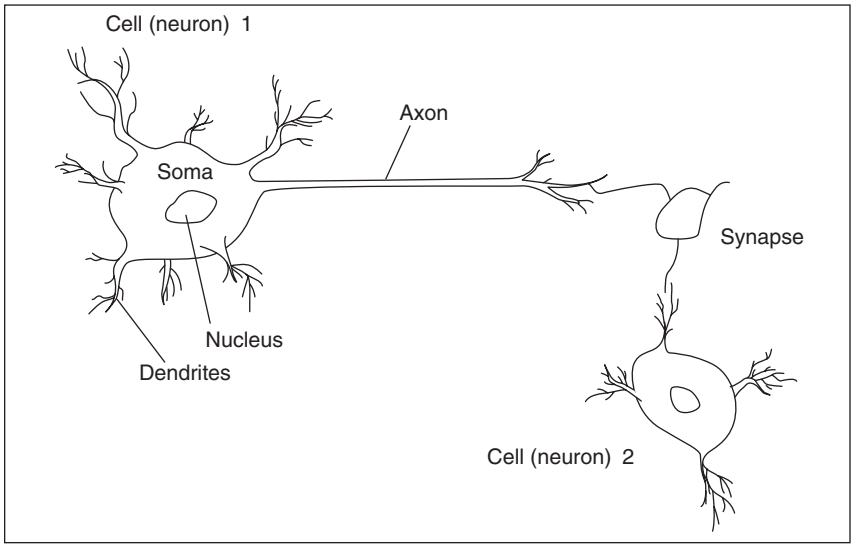


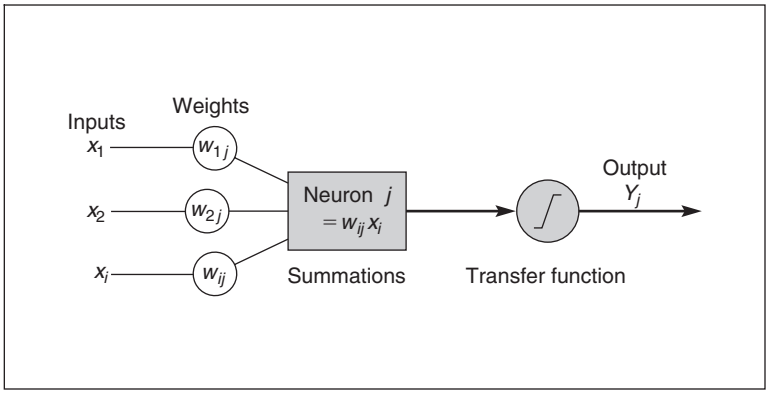
FIGURE 6.1 Portion of a Network: Two Interconnected Biological Cells

interconnected in a network architecture. The artificial neuron receives inputs analogous to the electrochemical impulses the dendrites of biological neurons receive from other neurons. The output of the artificial neuron corresponds to signals sent out from a biological neuron over its axon. These artificial signals can be changed by weights in a manner similar to the physical changes that occur in the synapses (see Figure 6.2).

Several ANN paradigms have been proposed for applications in a variety of problem domains. For example, see Application Case 6.2. Perhaps the easiest way to differentiate between the various models is on the basis of how these models structurally emulate the human brain, the way in which the neural model processes information and how the neural models learn to perform their designated tasks.

As they are biologically inspired, the main processing elements of a neural network are individual neurons, analogous to the brain's neurons. These artificial neurons

FIGURE 6.2 Processing Information in an Artificial Neuron



TECHNOLOGY INSIGHTS 6.1

The Relationship Between Biological and Artificial Neural Networks

The following list shows some of the relationships between biological and artificial networks:

Biological	Artificial
Soma	Node
Dendrites	Input
Axon	Output
Synapse	Weight
Slow speed	Fast speed
Many neurons (10^9)	Few neurons (a dozen to hundreds of thousands)

Zahedi (1993) talked about a dual role for ANN. We borrow concepts from the biological world to improve the design of computers. ANN technology is used for complex information processing and machine intelligence. On the other hand, neural networks can also be used as simple biological models to test hypotheses about "real" biological neuronal information processing. Of course, in the context of data mining, we are interested in the use of neural networks for machine learning and information processing.

Sources: L. Medsker and J. Liebowitz, *Design and Development of Expert Systems and Neural Networks*, Macmillan, New York, 1994, p. 163; and F. Zahedi, *Intelligent Systems for Business: Expert Systems with Neural Networks*, Wadsworth, Belmont, CA, 1993.

receive the sum "information" from other neurons or external input stimuli, perform a transformation on the inputs, and then pass on the transformed information to other neurons or external outputs. This is similar to how it is presently thought that the human brain works. Passing information from neuron to neuron can be thought of as a way to activate, or trigger a response from certain neurons based on the information or stimulus received.

Thus, how information is processed by a neural network is inherently a function of its structure. Neural networks can have one or more layers of neurons. These neurons can be highly or fully interconnected, or only certain layers can be connected together. Connections between neurons have an associated weight. In essence, the "knowledge" possessed by the network is encapsulated in these interconnection weights. Each neuron calculates a weighted sum of the incoming neuron values, transforms this input, and passes on its neural value as the input to subsequent neurons. Typically, although not always, this input/output transformation process at the individual neuron level is done in a nonlinear fashion.

Application Case 6.2

Neural Networks Help Reduce Telecommunications Fraud

The Forum of International Irregular Network Access (FIINA) estimates that telecommunications fraud results in a loss of US$55 billion per year worldwide. South Africa's largest telecom operator was losing over US$37 million per year to fraud. Subscription fraud—in which a customer either provides fraudulent details or gives valid details and then disappears—was the company's biggest cause of revenue leakage. By the time the telecom provider is alerted about the fraud, the fraudster has already moved to other target victims. Other types of fraud include phone card manipulation, which involves tampering and cloning of phone cards. In clip-on fraud, a fraudster

clips on to customers' telephone lines and then sells calls to overseas destinations for a fraction of normal rates.

Minotaur, developed by Neural Technologies (**neuralt.com**), was implemented to prevent fraud. Minotaur uses a hybrid mixture of intelligent systems and traditional computing techniques to provide customer subscription and real-time call monitoring fraud detection. It processes data from numerous fields, such as event data records (e.g., switch/CDR, SS#7, IPDRs, PIN/authentication) and customer data (e.g., billing and payment, point of sale, provisioning), using a multistream analysis capacity. Frauds are detected on several levels, such as on an individual basis using specific knowledge about the subscriber's usage, and on a global basis, using generic knowledge about subscriber usage and known fraud patterns. The neural capability of Minotaur means it learns from experience, making use of adaptive feedback to keep up-to-date with changing fraud patterns. A combination of call/network data and subscriber information is profiled and then processed, using intelligent neural, rule-based, and case-based techniques. Probable frauds are identified, collected into cases, and tracked to completion by means of a powerful and flexible workflow-based operational process.

In the first three months of installation of this neural network–based software:

- The average fraud loss per case was reduced by 40 percent.
- The detection time was reduced by 83 percent.
- The average time taken to analyze suspected fraud cases was reduced by 75 percent.
- The average detection hit rate was improved by 74 percent.

The combination of neural, rule-based, and case-based technologies provides a fraud detection rate superior to that of conventional systems. Furthermore, the multistream analysis capability makes it extremely accurate.

Sources: Combating Fraud: How a Leading Telecom Company Solved a Growing Problem, **neuralt.com/iqs/dlsfa.list/ dlcpti.7/downloads.html**; A. Shukla, *Neural Technologies and Sevis Partner to Eliminate Fraudulent Calls in Fixed and Mobile Networks*, February 3, 2006, **news.tmcnet.com/news/-neural-sevis-fraud-mobile/2006/02/03/1340423.htm** (accessed April 2006); P.A. Estévez, M.H. Claudio, and C.A. Perez, *Prevention in Telecommunications Using Fuzzy Rules and Neural Networks*, **cec.uchile.cl/~pestevez/RI0.pdf** (accessed April 2006); and *Members and Associate Members Success Stories*, **gsm.org/about/membership/success/nt.shtml** (accessed April 2006).

ELEMENTS OF ANN

A neural network is composed of processing elements organized in different ways to form the network's structure. The basic processing unit is the neuron. A number of neurons are organized into a network. There are many ways to organize neurons; they are referred to as **topologies**. One popular approach, known as the feedforward-backpropagation paradigm (or simply **backpropagation**), allows all neurons to link the output in one layer to the input of the next layer, but it does not allow any feedback linkage (Haykin, 1999). This is the most commonly used paradigm.

Processing Elements

The **processing elements (PE)** of an ANN are artificial neurons. Each of the neurons receives inputs, processes them, and delivers a single output, as shown in Figure 6.2. The input can be raw input data or the output of other processing elements. The output can be the final result (e.g., 1 means yes, 0 means no), or it can be inputs to other neurons.

Network Structure

Each ANN is composed of a collection of neurons, grouped in layers. A typical structure is shown in Figure 6.3. Note the three layers: *input*, *intermediate* (called the *hidden layer*), and *output*. A **hidden layer** is a layer of neurons that takes input from the previous layer and converts those inputs into outputs for further processing. Several hidden layers can be placed between the input and output layers, although it is quite common to use only one hidden layer. In that case, the hidden layer simply converts inputs into a nonlinear combination and passes the transformed inputs to the output layer. The most common interpretation of the hidden layer is as a feature extraction mechanism. That is, the hidden layer converts the original inputs in the problem into some higher-level combinations of such inputs.

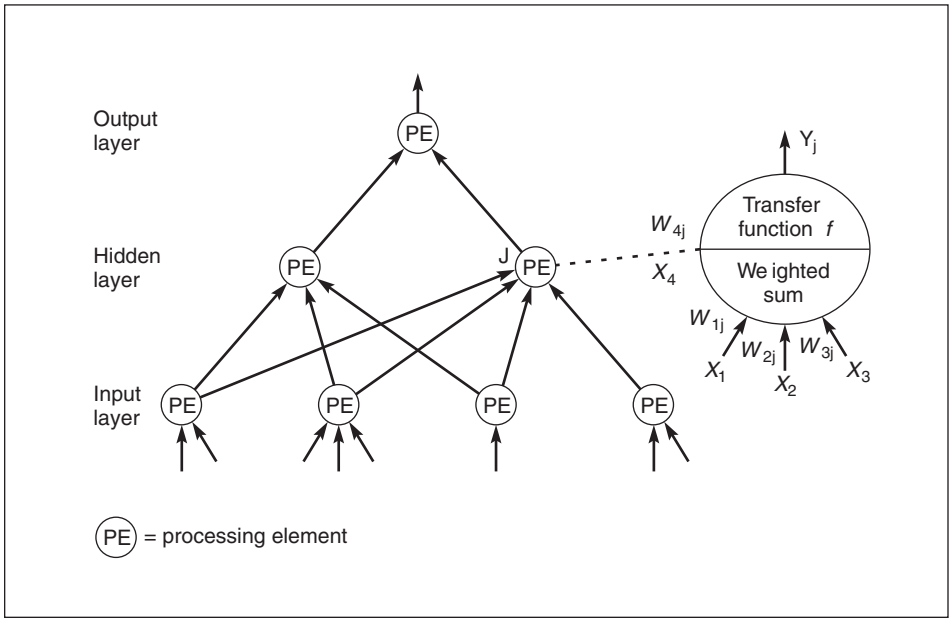


FIGURE 6.3 Neural Network with One Hidden Layer

Like a biological network, an ANN can be organized in several different ways (i.e., topologies or architectures); that is, the neurons can be interconnected in different ways. Therefore, ANN appear in many configurations called *architectures*. When information is processed, many of the processing elements perform their computations at the same time. This **parallel processing** resembles the way the brain works, and it differs from the serial processing of conventional computing.

Network Information Processing

Once the structure of a neural network is determined, information can be processed. We now present the major concepts related to the processing.

Inputs Each input corresponds to a single attribute. For example, if the problem is to decide on approval or disapproval of a loan, some attributes could be the applicant's income level, age, and home ownership. The numeric value, or representation, of an attribute is the input to the network. Several types of data, such as text, pictures, and voice, can be used as inputs. Preprocessing may be needed to convert the data to meaningful inputs from symbolic data or to scale the data.

Outputs The outputs of a network contain the solution to a problem. For example, in the case of a loan application, the outputs can be yes or no. The ANN assigns numeric values to the outputs, such as 1 for yes and 0 for no. The purpose of the network is to compute the values of the output. Often, postprocessing of the outputs is required because some networks use two outputs: one for yes and another for no. It is common to have to round the outputs to the nearest 0 or 1.

Connection Weights **Connection weights** are the key elements in an ANN. They express the relative strength (or mathematical value) of the input data or the many connections that transfer data from layer to layer. In other words, weights express the relative importance of each input to a processing element and, ultimately, the outputs.

Weights are crucial in that they store learned patterns of information. It is through repeated adjustments of weights that a network learns.

Summation Function The **summation function** computes the weighted sums of all the input elements entering each processing element. A summation function multiplies each input value by its weight and totals the values for a weighted sum *Y*. The formula for *n* inputs in one processing element (see Figure 6.4a) is:

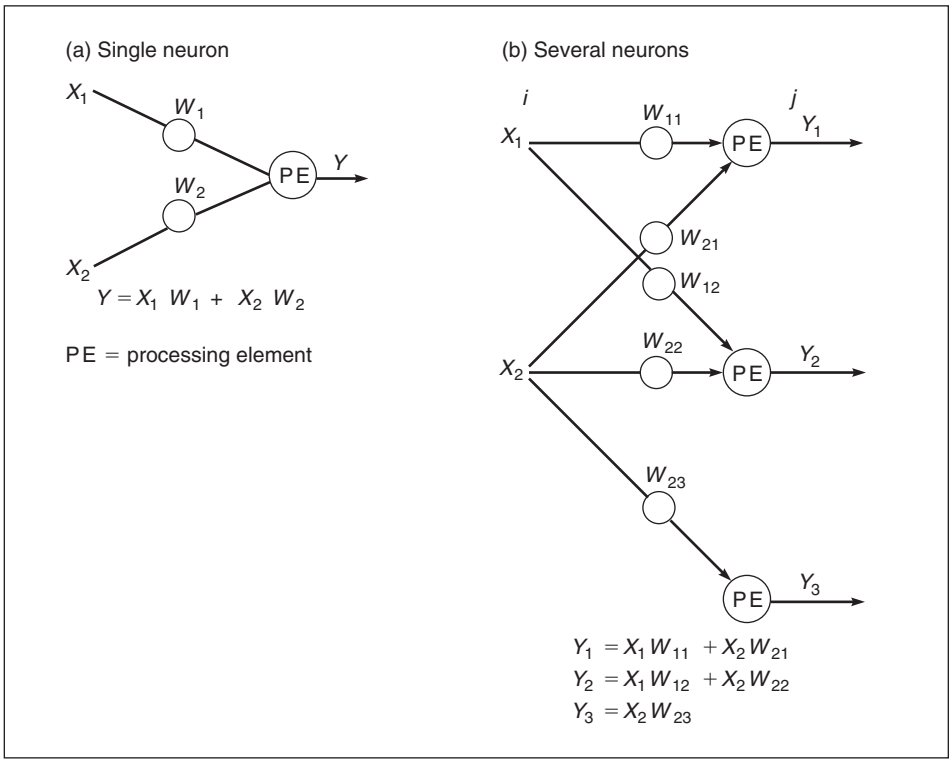
$$Y = \sum_{i=1}^{n} X_i W_i$$

For the *j*th neuron of several processing neurons in a layer (see Figure 6.4b), the formula is:

$$Yj = \sum_{i=1}^{n} X_i W_{ij}$$

Transformation (Transfer) Function The summation function computes the internal stimulation, or activation level, of the neuron. Based on this level, the neuron may or may not produce an output. The relationship between the internal activation level and the output can be linear or nonlinear. The relationship is expressed by one

FIGURE 6.4 Summation Function for a Single Neuron (a) and Several Neurons (b)



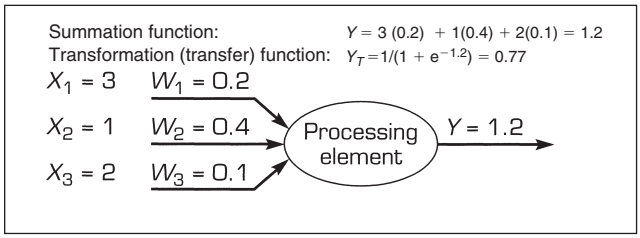


FIGURE 6.5 Example of ANN Functions

of several types of *transformation (transfer) functions*. The **transformation (transfer) function** combines (i.e., adds up) the inputs coming into a neuron from other neurons/sources and then produces an output based on the choice of the transfer function. Selection of the specific function affects the network's operation. The **sigmoid (logical activation) function** (or sigmoid *transfer function*) is an S-shaped transfer function in the range of 0 to 1, and it is a popular as well as useful nonlinear transfer function:

$$Y_T = 1/(1 + e^{-Y})$$

where Y_T is the transformed (i.e., normalized) value of Y (see Figure 6.5).

The transformation modifies the output levels to reasonable values (typically between 0 and 1). This transformation is performed before the output reaches the next level. Without such a transformation, the value of the output becomes very large, especially when there are several layers of neurons. Sometimes, instead of a transformation function, a *threshold value* is used. A **threshold value** is a hurdle value for the output of a neuron to trigger the next level of neurons. If an output value is smaller than the threshold value, it will not be passed to the next level of neurons. For example, any value of 0.5 or less becomes 0, and any value above 0.5 becomes 1. A transformation can occur at the output of each processing element, or it can be performed only at the final output nodes.

Hidden Layers

Complex practical applications require one or more hidden layers between the input and output neurons and a correspondingly large number of weights. Many commercial ANN include three and sometimes up to five layers, with each containing 10 to 1,000 processing elements. Some experimental ANN use millions of processing elements. Because each layer increases the training effort exponentially and also increases the computation required, the use of more than three hidden layers is rare in most commercial systems.

NEURAL NETWORK ARCHITECTURES

There are several effective neural network models and algorithms (see Haykin, 1999). Some of the most common are *backpropagation (*or *feedforward), associative memory*, and the *recurrent network*. The backpropagation architecture is shown in Figure 6.3. The other two architectures are shown in Figures 6.6 and 6.7.

Ultimately, the operation of the entire neural network model is driven by the task it is designed to address. For instance, neural network models have been used as

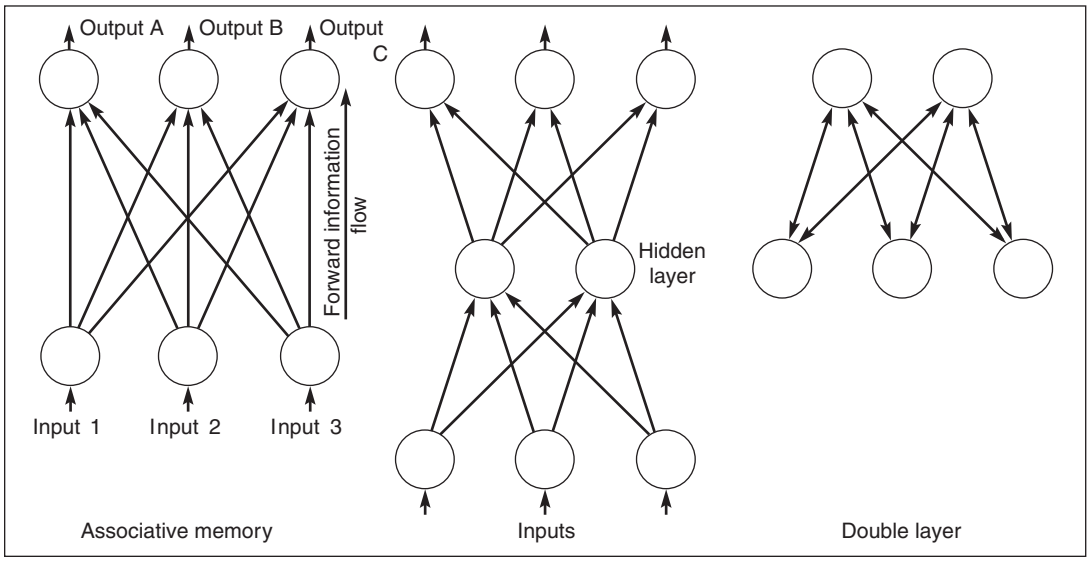


FIGURE 6.6 Neural Network Structures: Feedforward Flow

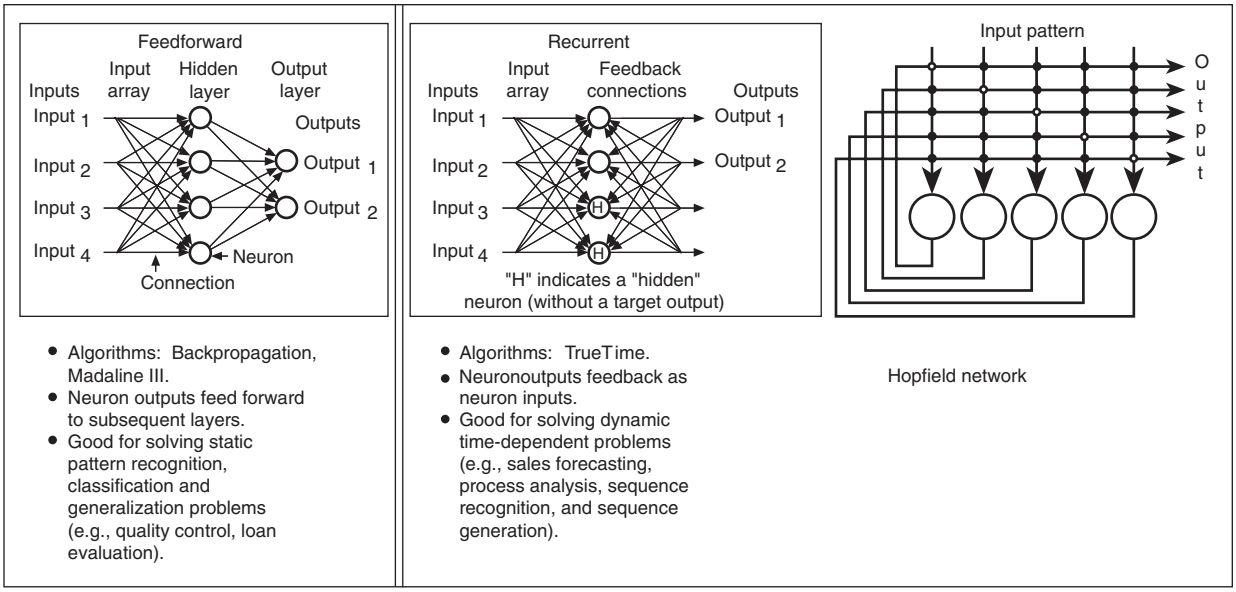


FIGURE 6.7 Recurrent Structure Compared with Feedforward Source

Source: Based on *PC AI*, May/June 1992, p. 35.

classifiers, as forecasting tools, and as general optimizers. As shown later in this chapter, neural network classifiers are typically multilayer models in which information is passed from one layer to the next, with the ultimate goal of mapping an input to the network to a specific category, as identified by an output of the network. A neural

model used as an optimizer, on the other hand, can be a single layer of neurons, highly interconnected, and can compute neuron values iteratively until the model converges to a stable state. This stable state would then represent an optimal solution to the problem under analysis.

Finally, how a network is trained to perform its desired task is another identifying model characteristic. Neural network learning can occur in either a supervised or unsupervised mode. In supervised learning, a sample training set is used to "teach" the network about its problem domain. This training set of exemplar cases (input and the desired output[s]) is iteratively presented to the neural network. Output of the network in its present form is calculated and compared to the desired output. The **learning algorithm** is the training procedure that an ANN uses. The learning algorithm being used determines how the neural interconnection weights are corrected due to differences in the actual and desired output for a member of the training set. Updating of the network's interconnection weights continues until the stopping criteria of the training algorithm are met (e.g., all cases must be correctly classified within a certain tolerance level).

Alternatively, in unsupervised learning, there are no target answers that the network tries to learn. Instead, the neural network learns a pattern through repeated exposure. Thus, this kind of learning can be envisioned as the neural network appropriately self-organizing or clustering its neurons related to the specific desired task.

Multilayer, feedforward neural networks are a class of models that show promise in classification and forecasting problems. As the name implies, these models structurally consist of multiple layers of neurons. Information is passed through the network in one direction, from the input layers of the network, through one or more hidden layers, toward the output layer of neurons. Neurons of each layer are connected only to the neurons of the subsequent layer.

Section 6.2 Review Questions

1. What is an ANN?
2. Explain the following terms: *neuron*, *axon*, and *synapse*.
3. How do weights function in an ANN?
4. What is the role of the summation function?
5. What is the role of the transformation function?

6.3 LEARNING IN ANN

An important consideration in an ANN is the use of an appropriate learning algorithm (or *training algorithm*). Learning algorithms specify the process by which a neural network learns the underlying relationship between input and outputs, or just among the inputs. There are hundreds of them. Learning algorithms in ANN can also be classified as supervised learning and unsupervised learning (see Figure 6.8).

Supervised learning uses a set of inputs for which the appropriate (i.e., desired) outputs are known. For example, a historical set of loan applications with the success or failure of the individual to repay the loan has a set of input parameters and presumed known outputs. In one type, the difference between the desired and actual outputs is used to calculate corrections to the weights of the neural network. A variation of this

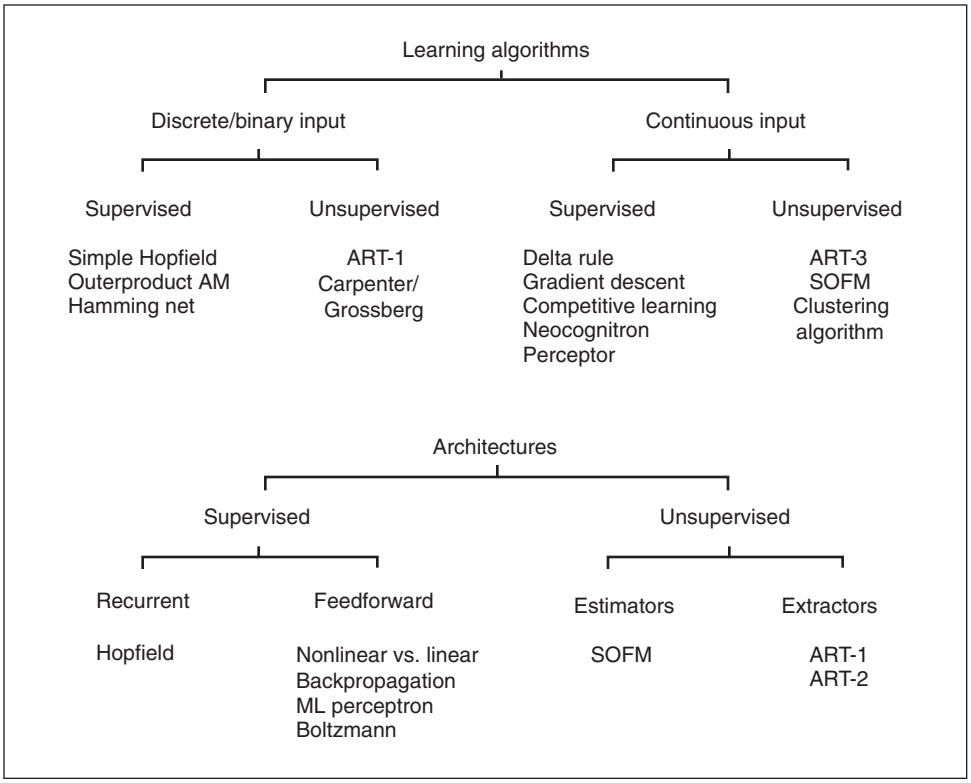


FIGURE 6.8 Taxonomy of ANN Architectures and Learning Algorithms

Source: Based on L. Medsker and J. Liebowitz, *Design and Development of Expert Systems and Neural Computing,* Macmillan, New York, 1994, p. 166.

approach simply acknowledges for each input trial whether the output is correct as the network adjusts weights in an attempt to achieve correct results. Examples of this type of learning are backpropagation and the Hopfield network (Hopfield, 1982). Application Case 6.3 illustrates an application of supervised learning at Microsoft for improving the response rate to target mailings to potential customers.

Application Case 6.3

Neural Networks Help Deliver Microsoft's Mail to the Intended Audience

Microsoft, the world leader in computer software, based in Redmond, Washington, uses BrainMaker neural network software from California Scientific (**calsci.com**) to maximize returns on direct mail. Every year, Microsoft sends about 40 million pieces of direct mail to 8.5 million registered customers, aimed at encouraging people to upgrade their software or to buy other related products. Generally, the first mailing includes everyone in the database. The key is to direct the second mailing to only those who are most likely to respond.

Several variables were fed into the BrainMaker neural network to get productive results. The first step was to identify the variables that were relevant and to eliminate the variables that did not cause any effect. The following were some of the significant variables:

- Recency, calculated in number of days, which measures the last time something was bought and registered. It is likely that the more recently a customer has bought something, the better the chance that he or she is going to buy more.
- First date to file, which is the date an individual made his or her first purchase. This is a measure of loyalty. Chances are high that a customer will buy again if he or she has been a loyal customer.
- The number of products bought and registered.
- The value of the products bought and registered, calculated at the standard reselling price.
- The number of days between the time the product came out and when it was purchased; research has shown that people who tend to buy things as soon as they are available are the key individuals to be reached.

Several other personal characteristics were also added and scored with yes/no responses.

Before training, the information obtained from the customer responses was fed into a format the network could use, and yes/no responses were transformed to numeric data. Minimums and maximums were set on certain variables.

Initially, the network was trained with 25 variables. The data were taken from seven or eight campaigns to make it varied and represent all aspects of the business, including the Mac and Windows sides, from high and low price-point products.

Before Microsoft began using BrainMaker, an average mailing would get a response rate of 4.9 percent. By using BrainMaker, the response rate has increased to 8.2 percent. The neural network was tested on data from 20 different campaigns with known results not used during training. The results showed repeated and consistent savings. An average mailing resulted in a 35 percent cost savings for Microsoft.

Sources: California Scientific, "Maximize Returns on Direct Mail with BrainMaker Neural Networks Software," **calsci.com/DirectMail.html**; and G. Piatesky-Shapiro, *ISR:Microsoft Success Using Neural Network for Direct Marketing,* **Kdnuggets.com/news/94/n9.txt** (accessed March 2006).

In **unsupervised learning**, only input stimuli are shown to the network. The network is **self-organizing**; that is, it organizes itself internally so that each hidden processing element responds strategically to a different set of input stimuli (or groups of stimuli). No knowledge is supplied about which classifications (i.e., outputs) are correct, and those that the network derives may or may not be meaningful to the network developer (this is useful for cluster analysis). However, by setting model parameters, we can control the number of categories into which a network classifies the inputs. Regardless, a human must examine the final categories to assign meaning and determine the usefulness of the results. Examples of this type of learning are **adaptive resonance theory (ART)** (i.e., a neural network architecture that is aimed at being brain-like in unsupervised mode) and **Kohonen self-organizing feature maps** (i.e., neural network models for machine learning).

As mentioned earlier, many different and distinct neural network paradigms have been proposed for various decision-making domains. A neural model that has been shown appropriate for classification problems (e.g., bankruptcy prediction) is the feedforward MLP. Multilayered networks have continuously valued neurons (i.e., processing elements), are trained in a supervised manner, and consist of one or more layers of nodes (i.e., hidden nodes) between the input and output nodes. A typical feedforward neural network is shown in Figure 6.3. Input nodes represent where information is presented to the network, output nodes provide the "decision" of the neural network, and the hidden nodes via the interconnection weights contain the proper mapping of inputs to outputs (i.e., decisions).

The backpropagation learning algorithm is the standard way of implementing supervised training of feedforward neural networks. It is an iterative gradient-descent technique designed to minimize an error function between the actual output of the network and its desired output, as specified in the training set of data. Adjustment of the interconnection weights, which contain the mapping function per se, starts at the output node where the error measure is initially calculated and is then propagated

back through the layers of the network, toward the input layer. More details are included in the following section.

THE GENERAL ANN LEARNING PROCESS

In supervised learning, the learning process is inductive; that is, connection weights are derived from existing cases. The usual process of learning involves three tasks (see Figure 6.9):

1. Compute temporary outputs.
2. Compare outputs with desired targets.
3. Adjust the weights and repeat the process.

When existing outputs are available for comparison, the learning process starts by setting the connection weights, either via rules or randomly. The difference between the actual output (Y or Y_T) and the desired output (Z) for a given set of inputs is an error called *delta* (in calculus, the Greek symbol delta, Δ, means "difference").

The objective is to minimize the delta (i.e., reduce it to 0 if possible), which is done by adjusting the network's weights. The key is to change the weights in the right direction, making changes that reduce the delta (i.e., error). We will show how this is done later.

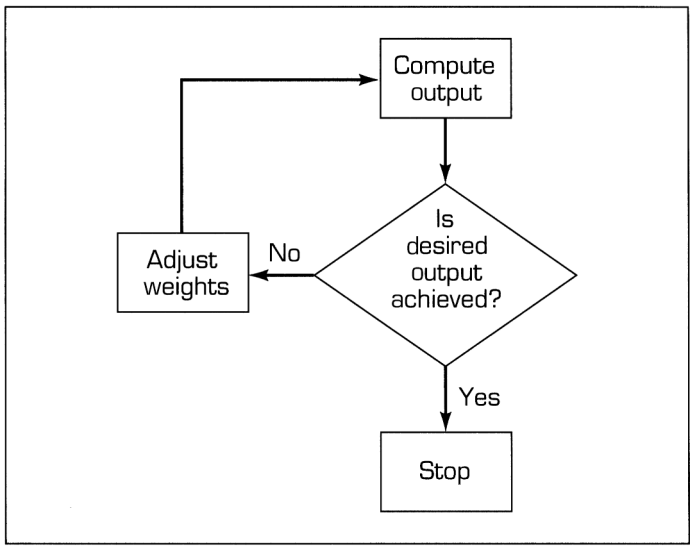
Information processing with an ANN consists of an attempt to recognize patterns of activities (i.e., **pattern recognition**). During the learning stages, the interconnection weights change in response to training data presented to the system.

Different ANN compute delta in different ways, depending on the learning algorithm being used. There are hundreds of learning algorithms for various situations and configurations, some of which are discussed later in this chapter.

HOW A NETWORK LEARNS

Consider a single neuron that learns the inclusive OR operation—a classic problem in symbolic logic. There are two input elements, X_1 and X_2. If either or both of them have a positive value, the result is also positive. This can be shown as follows:

FIGURE 6.9 Learning Process of an ANN



	Inputs		
Case	X_1	X_2	***Desired Results***
1	0	0	0
2	0	1	1 (positive)
3	1	0	1 (positive)
4	1	1	1 (positive)

The neuron must be trained to recognize the input patterns and classify them to give the corresponding outputs. The procedure is to present to the neuron the sequence of the four input patterns so that the weights are adjusted after each iteration (using feedback of the error found by comparing the estimate to the desired result). This step is repeated until the weights converge to a uniform set of values that allow the neuron to classify each of the four inputs correctly. The results shown in Table 6.1 were produced in Excel. In this simple example, a threshold function is used to evaluate the summation of input values. After calculating outputs, a measure of the error (i.e., delta) between the output and the desired values is used to update the weights, subsequently reinforcing the correct results. At any step in the process for a neuron j we have:

$$\text{delta} = Z_j - Y_j$$

where Z and Y are the desired and actual outputs, respectively. Then, the updated weights are:

$$W_i(\text{final}) = W_i(\text{initial}) + \text{alpha} \times \text{delta} \times X_i$$

where alpha is a parameter that controls how fast the learning takes place. This is called a **learning rate**. The choice of the learning rate parameter can have an impact on how fast (and how correctly) a neural network learns. A high value for the learning rate can

TABLE 6.1 Example of Supervised Learning[a]

				Initial				***Final***	
Step	X_1	X_2	Z	W_1	W_2	Y	***Delta***	W_1	W_2
1	0	0	0	0.1	0.3	0	0.0	0.1	0.3
	0	1	1	0.1	0.3	0	1.0	0.1	0.5
	1	0	1	0.1	0.5	0	1.0	0.3	0.5
	1	1	1	0.3	0.5	1	0.0	0.3	0.5
2	0	0	0	0.3	0.5	0	0.0	0.3	0.5
	0	1	1	0.3	0.5	0	0.0	0.3	0.7
	1	0	1	0.3	0.7	0	1.0	0.5	0.7
	1	1	1	0.5	0.7	1	0.0	0.5	0.7
3	0	0	0	0.5	0.7	0	0.0	0.5	0.7
	0	1	1	0.5	0.7	1	0.0	0.5	0.7
	1	0	1	0.5	0.7	0	1.0	0.7	0.7
	1	1	1	0.7	0.7	1	0.0	0.7	0.7
4	0	0	0	0.7	0.7	0	0.0	0.7	0.7
	0	1	1	0.7	0.7	1	0.0	0.7	0.7
	1	0	1	0.7	0.7	1	0.0	0.7	0.7
	1	1	1	0.7	0.7	1	0.0	0.7	0.7

[a]Parameters: alpha = 0.2; threshold = 0.5, output is zero if the sum (W_1*X_1+W_2*X_2) is not greater than 0.5.

lead to too much correction in the weight values, resulting in going back and forth among possible weights values and never reaching the optimal, which may lie somewhere in between the endpoints. Too low a learning rate may slow down the learning process. In practice, a neural network analyst may try using many different choices of learning rates to achieve optimal learning.

Most implementations of the learning process also include a counterbalancing parameter called *momentum* to provide a balance to the learning rate. Essentially, whereas learning rate is aimed at correcting for the error, **momentum** is aimed at slowing down the learning. Many of the software programs available for neural networks today can automatically select these parameters for the user or let the user experiment with many different combinations of such parameters.

As shown in Table 6.1, each calculation uses one of the X_1 and X_2 pairs and the corresponding value for the OR operation, along with the initial values W_1 and W_2 of the neuron's weights. Initially, the weights are assigned random values, and the learning rate, alpha, is set low. Delta is used to derive the final weights, which then become the initial weights in the next iteration (i.e., row).

The initial values of weights for each input are transformed using the equation shown earlier to assign the values that are used with the next input (i.e., row). The threshold value (0.5) sets the output Y to 1 in the next row if the weighted sum of inputs is greater than 0.5; otherwise, Y is set to 0. In the first step, two of the four outputs are incorrect (delta = 1), and a consistent set of weights has not been found. In subsequent steps, the learning algorithm improves the results, until it finally produces a set of weights that give the correct results ($W_1 = W_2 = 0.7$ in step 4 of Table 6.1). Once determined, a neuron with these weight values can quickly perform the OR operation.

In developing an ANN, an attempt is made to fit the problem characteristic to one of the known learning algorithms. There are software programs for all the different algorithms, such as backpropagation, which we describe next. Many variants of this algorithm exist, but the core concepts of them all are similar.

BACKPROPAGATION

Backpropagation (short for *back-error propagation*) is the most widely used supervised learning algorithm in neural computing (Principe et al., 2000). It is very easy to implement. A backpropagation network includes one or more hidden layers. This type of network is considered feedforward because there are no interconnections between the output of a processing element and the input of a node in the same layer or in a preceding layer. Externally provided correct patterns are compared with the neural network's output during (supervised) training, and feedback is used to adjust the weights until the network has categorized all the training patterns as correctly as possible (the error tolerance is set in advance).

Starting with the output layer, errors between the actual and desired outputs are used to correct the weights for the connections to the previous layer (see Figure 6.10). For any output neuron j, the error (delta) = $(Z_j - Y_j)(df/dx)$, where Z and Y are the desired and actual outputs, respectively. Using the sigmoid function, $f = [1 + \exp(-x)]^{-1}$, where x is proportional to the sum of the weighted inputs to the neuron, is an effective way to compute the output of a neuron in practice. With this function, the derivative of the sigmoid function $df/dx = f(1 - f)$ and the error is a simple function of the desired and actual outputs. The factor $f(1 - f)$ is the *logistic function*, which serves to keep the error correction well bounded. The weights of each input to the jth neuron are then changed in proportion to this calculated error. A more complicated expression can be derived to work backward in a similar way from the output neurons through the hidden

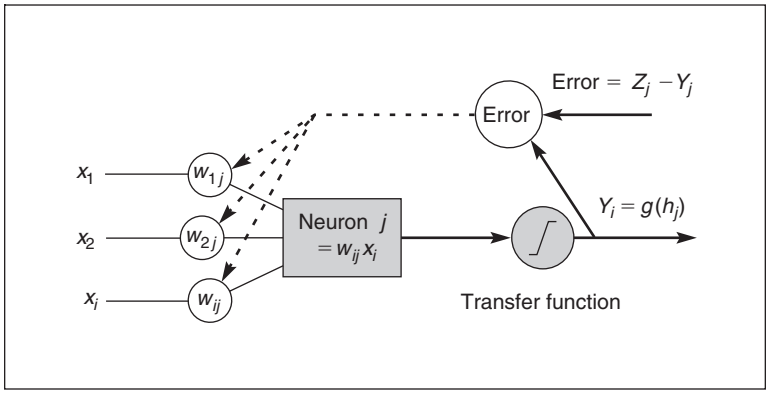


FIGURE 6.10 Backpropagation of Errors for a Single Neuron

layers to calculate the corrections to the associated weights of the inner neurons. This complicated method is an iterative approach to solving a nonlinear optimization problem that is very similar in meaning to the one characterizing multiple-linear regression.

The learning algorithm includes the following procedures:

1. Initialize weights with random values and set other parameters.
2. Read in the input vector and the desired output.
3. Compute the actual output via the calculations, working forward through the layers.
4. Compute the error.
5. Change the weights by working backward from the output layer through the hidden layers.

This procedure is repeated for the entire set of input vectors until the desired output and the actual output agree within some predetermined tolerance. Given the calculation requirements for one iteration, a large network can take a very long time to train; therefore, in one variation, a set of cases are run forward and an aggregated error is fed backward to speed up learning. Sometimes, depending on the initial random weights and network parameters, the network does not converge to a satisfactory performance level. When this is the case, new random weights must be generated, and the network parameters, or even its structure, may have to be modified before another attempt is made. Current research is aimed at developing algorithms and using parallel computers to improve this process. For example, genetic algorithms can be used to guide the selection of the network structure, as mentioned in the opening vignette.

Section 6.3 Review Questions

1. Briefly describe backpropagation.
2. What is the purpose of a threshold value in a learning algorithm?
3. What is the purpose of a learning rate?
4. How does error between actual and predicted outcomes affect the value of weights in neural networks?

6.4 DEVELOPING NEURAL NETWORK–BASED SYSTEMS

Although the development process of ANN is similar to the structured design methodologies of traditional computer-based information systems, some phases are unique or have some unique aspects. In the process described here, we assume that the preliminary steps of system development, such as determining information requirements, conducting a feasibility analysis, and gaining a champion in top management for the project, have been completed successfully. Such steps are generic to any information system.

As shown in Figure 6.11, the development process for an ANN application includes nine steps. In step 1, the data to be used for training and testing the network are collected. Important considerations are that the particular problem is amenable to neural network solution and that adequate data exist and can be obtained. In step 2, training data must be identified, and a plan must be made for testing the performance of the network.

In steps 3 and 4, a network architecture and a learning method are selected. The availability of a particular development tool or the capabilities of the development personnel may determine the type of neural network to be constructed. Also, certain problem types have demonstrated high success rates with certain configurations (e.g., multilayer feedforward neural networks for bankruptcy prediction, as described in the next section). Important considerations are the exact number of neurons and the number of layers. Some packages use genetic algorithms to select the network design.

There are parameters for tuning the network to the desired learning-performance level. Part of the process in step 5 is the initialization of the network weights and parameters, followed by the modification of the parameters as training-performance feedback is received. Often, the initial values are important in determining the efficiency and length of training. Some methods change the parameters during training to enhance performance.

Step 6 transforms the application data into the type and format required by the neural network. This may require writing software to preprocess the data or performing these operations directly in an ANN package. Data storage and manipulation techniques and processes must be designed for conveniently and efficiently retraining the neural network, when needed. The application data representation and ordering often influence the efficiency and possibly the accuracy of the results.

In steps 7 and 8, training and testing are conducted iteratively by presenting input and desired or known output data to the network. The network computes the outputs and adjusts the weights until the computed outputs are within an acceptable tolerance of the known outputs for the input cases. The desired outputs and their relationships to input data are derived from historical data (i.e., a portion of the data collected in step 1).

In step 9, a stable set of weights is obtained. Now the network can reproduce the desired outputs, given inputs such as those in the training set. The network is ready for use as a standalone system or as part of another software system where new input data will be presented to it and its output will be a recommended decision.

In the following sections, we examine these steps in more detail.

DATA COLLECTION AND PREPARATION

The first two steps in the ANN development process involve collecting data and separating them into a training set and a testing set. The training cases are used to adjust the weights, and the testing cases are used for network validation. The data used for training

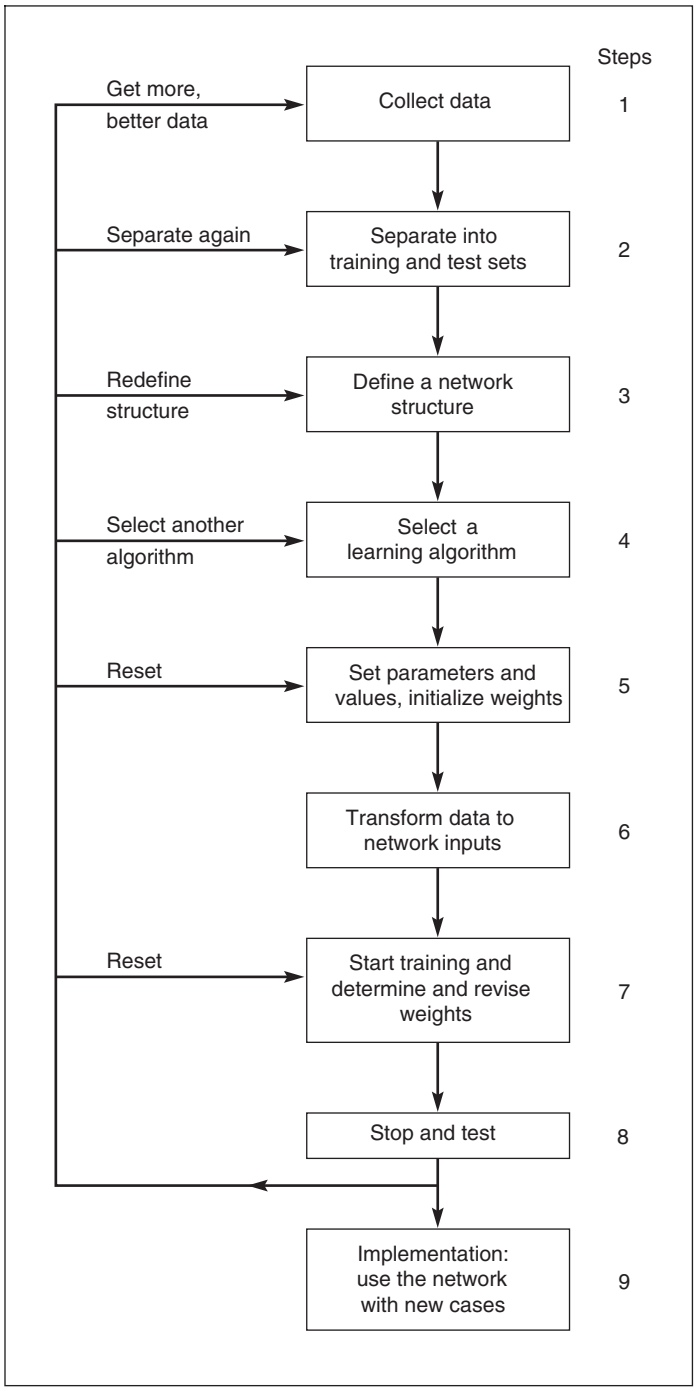


FIGURE 6.11 Flow Diagram of the Development Process of an ANN

and testing must include all the attributes that are useful for solving the problem. The system can only learn as much as the data can tell. Therefore, collection and preparation of data is the most critical step in building a good system.

In general, the more data used, the better. Larger data sets increase processing time during training but improve the accuracy of the training and often lead to faster convergence to a good set of weights. For a moderately sized data set, typically 80 percent of the data are randomly selected for training and 20 percent are selected for testing; for small data sets, typically all the data are used for training and testing; and for large data sets, a sufficiently large sample is taken and treated like a moderately sized data set.

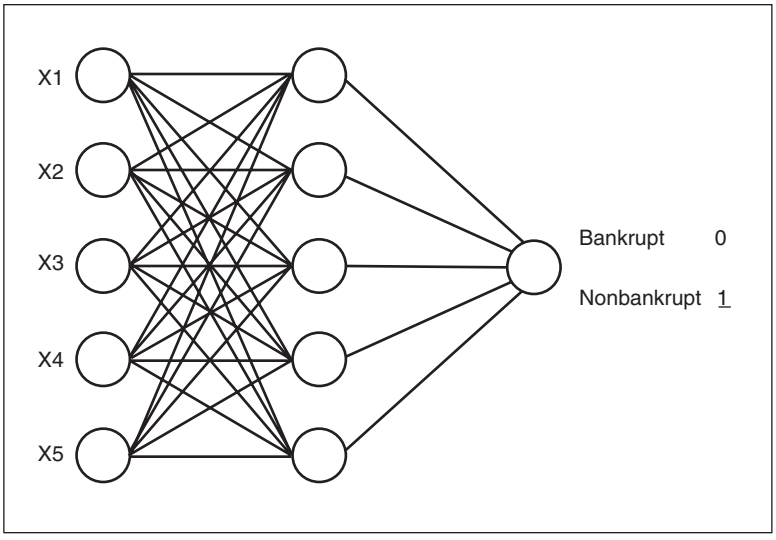
For example, say a bank wants to build a neural network–based system in order to use clients' financial data to determine whether they may go bankrupt. The bank needs to first identify what financial data may be used as inputs and how to obtain them. Five attributes may be useful inputs: (1) working capital/total assets, (2) retained earnings/total assets, (3) earnings before interest and taxes/total assets, (4) market value of equity/total debt, and (5) sales/total sales. The output is a binary variable: bankruptcy or not.

SELECTION OF NETWORK STRUCTURE

After the training and testing data sets are identified, the next step is to design the structure of the neural networks. This includes the selection of a topology and determination of (1) input nodes, (2) output nodes, (3) number of hidden layers, and (4) number of hidden nodes. The multilayer feedforward topology is often used in business applications, although other network models are beginning to find some business use as well.

The design of input nodes must be based on the attributes of the data set. In the example of predicting bankruptcy, for example, the bank might choose a three-layer structure that includes one input layer, one output layer, and one hidden layer. The input layer contains five nodes, each of which is a variable, and the output layer contains a node with 0 for bankrupt and 1 for safe. Determining the number of hidden nodes is tricky. A few heuristics have been proposed, but none of them is unquestionably the best. A typical approach is to choose the average number of input and output nodes. In the previous case, the hidden node may be set to $(5 + 1)/2 = 3$. Figure 6.12 shows a possible structure for the bankruptcy-prediction problem.

FIGURE 6.12 Architecture of the Bankruptcy Prediction Neural Network



LEARNING ALGORITHM SELECTION

After the network structure is chosen, we need to find a learning algorithm to identify a set of connection weights that best cover the training data and have the best predictive accuracy. For the feedforward topology we chose for the bankruptcy-prediction problem, a typical approach is to use the backpropagation algorithm. Because many commercial packages are available on the market, there is no need to implement the learning algorithm by ourselves. Instead, we can choose a suitable commercial package to analyze the data. Technology Insights 6.4 summarizes information on different types of neural network software packages that are available.

NETWORK TRAINING

Training of ANN is an iterative process that starts from a random set of weights and gradually enhances the fitness of the network model and the known data set. The iteration continues until the error sum is converged to below a preset acceptable level. In the

TECHNOLOGY INSIGHTS 6.4

ANN Software

There are many tools for developing neural networks (see this book's Web site and the periodic resource lists in *PC AI*, **pcai.com**). Some of these tools function like expert system shells. They provide a set of standard architectures, learning algorithms, and parameters, along with the ability to manipulate the data. Some development tools can support up to several dozen network paradigms and learning algorithms.

Neural network implementations are also available in most of the comprehensive data mining tools, such as the SAS Enterprise Miner, Clementine, and STATISTICA Data Miner. WEKA is an open source collection of machine learning algorithms for data mining tasks, and it includes neural network capabilities. WEKA can be downloaded from **cs.waikato.ac.nz/~ml/weka**. STATISTICA is available on trial basis to adopters of this book.

Many specialized neural network tools enable the building and deployment of a neural network model in practice. Any listing of such tools would be incomplete. Online resources such as Wikipedia (**en.wikipedia.org/wiki/Artificial_neural_network**), the Google or Yahoo software directory, and vendor listings on **pcai.com** are good places to locate the latest information on neural network software vendors. Some of the vendors that have been around for a while and have reported industrial applications of their neural network software include California Scientific (BrainMaker), NeuralWare, NeuroDimension Inc., Ward Systems Group (Neuroshell), and Megaputer. Again, the list can never be complete.

Some ANN development tools are spreadsheet add-ins. Most can read spreadsheet, database, and text files. Some are freeware or shareware. Some ANN systems have been developed in Java to run directly on the Web and are accessible through a Web browser interface. Other ANN products are designed to interface with expert systems as hybrid development products.

Developers may instead prefer to use more general programming languages, such as C++, or a spreadsheet to program the model and perform the calculations. A variation on this is to use a library of ANN routines. For example, hav.Software (**hav.com**) provides a library of C++ classes for implementing standalone or embedded feedforward, simple recurrent, and random-order recurrent neural networks. Computational software such as MATLAB also includes neural network–specific libraries.

How are neural networks implemented in practice? After the analyst/developer has conducted enough tests to ascertain that a neural network can do a good job for the application, the network needs to be implemented in the existing systems. A number of neural network shells can generate code, in C++, Java, or Visual Basic, that can be embedded in another system that can access source data or is called directly by a graphical user interface for deployment, independently of the development system. Or, after training an ANN in a development tool, given the weights, network structure, and transfer function, one can easily develop one's own implementation in a third-generation programming language such as C++. Most of the ANN development packages as well as data mining tools can generate such code. The code can then be embedded in a standalone application or in a Web server application.

backpropagation algorithm, two parameters, *learning rate* and *momentum*, can be adjusted to control the speed of reaching a solution. These determine the ratio of the difference between the calculated value and the actual value of the training cases. Some software packages may have their own parameters in their learning heuristics to speed up the learning process. It is important to read carefully when using this type of software.

Some data conversion may be necessary in the training process. This includes (1) changing the data format to meet the requirements of the software, (2) normalizing the data scale to make the data more comparable, and (3) removing problematic data. When the training data set is ready, it is loaded into the package, and the learning procedure is executed. Depending on the number of nodes and the size of the training data set, reaching a solution may take from a few thousand to millions of iterations.

TESTING

Recall that in step 2 of the development process shown in Figure 6.11, the available data are divided into training and testing data sets. When the training has been completed, it is necessary to test the network. Testing (step 8) examines the performance of the derived network model by measuring its ability to classify the testing data correctly. **Black-box testing** (i.e., comparing test results to historical results) is the primary approach for verifying that inputs produce the appropriate outputs. Error terms can be used to compare results against known benchmark methods.

The network is generally not expected to perform perfectly (zero error is difficult, if not impossible, to attain), and only a certain level of accuracy is really required. For example, if 1 means nonbankrupt and 0 means bankrupt, then any output between 0.1 and 1 might indicate a certain likelihood of nonbankrupty. The neural network application is usually an alternative to another method that can be used as a benchmark against which to compare accuracy. For example, a statistical technique such as multiple regression or another quantitative method may be known to classify inputs correctly 50 percent of the time.

The neural network implementation often improves on this. For example, Liang (1992) reported that ANN performance was superior to the performance of multiple discriminant analysis and rule induction. Ainscough and Aronson (1999) investigated the application of neural network models in predicting retail sales, given a set of several inputs (e.g., regular price, various promotions). They compared their results to those of multiple regression and improved the adjusted R^2 (correlation coefficient) from .5 to .7. If the neural network is replacing manual operations, performance levels and speed of human processing can be the standard for deciding whether the testing phase is successful.

The test plan should include routine cases as well as potentially problematic situations. If the testing reveals large deviations, the training set must be reexamined, and the training process may have to be repeated (some "bad" data may have to be omitted from the input set).

Note that we cannot equate neural network results exactly with those found using statistical methods. For example, in stepwise linear regression, input variables are sometimes determined to be insignificant, but because of the nature of neural computing, a neural network uses them to attain higher levels of accuracy. When they are omitted from a neural network model, its performance typically suffers.

IMPLEMENTATION OF AN ANN

Implementation of an ANN (step 9) often requires interfaces with other computer-based information systems and user training. Ongoing monitoring and feedback to the developers are recommended for system improvements and long-term success. It is

also important to gain the confidence of users and management early in the deployment to ensure that the system is accepted and used properly.

Section 6.4 Review Questions

1. List the nine steps in conducting a neural network project.
2. What are some of the design parameters for developing a neural network?
3. Describe different types of neural network software available today.
4. How are neural networks implemented in practice when the training/testing is complete?
5. What parameters may need to be adjusted in the neural network training process?

6.5 A SAMPLE NEURAL NETWORK PROJECT

We next describe a typical application of neural networks to predict bankruptcy of companies using the same data and a similar experimental design as used by Wilson and Sharda (1994). For comparative purposes, the performance of neural networks is contrasted with logistic regression.

The Altman (1968) study has been used as the standard of comparison for many bankruptcy classification studies using discriminant analysis and logistic regression; follow-up studies have identified several other attributes to improve prediction performance. We use the same financial ratios as in Altman's study, realizing that more sophisticated inputs to the neural network model should only enhance its performance. These ratios are as follows:

X_1: Working capital/total assets
X_2: Retained earnings/total assets
X_3: Earnings before interest and taxes/total assets
X_4: Market value of equity/total debt
X_5: Sales/total assets

Step 1 consists of collecting relevant data. The sample of firms for which these ratios was obtained from *Moody's Industrial Manuals*. It consisted of firms that either were in operation or went bankrupt between 1975 and 1982. The sample consists of a total of 129 firms, 65 of which went bankrupt during the period and 64 nonbankrupt firms matched on industry and year. Data used for the bankrupt firms are from the last financial statements issued before the firms declared bankruptcy. Thus, the prediction of bankruptcy is to be made about 1 year in advance.

Step 2 requires us to break the data set into a training set and a testing set. Because the determination of the split may affect experimental findings, a resampling procedure can be used to create many different pairs of training and testing sets, which also ensures that there is no overlap in the composition of the matched training and testing sets. For example, a training set of 20 patterns can be created by randomly setting 20 records from the collected set. A set of 20 other patterns/records can be created as a test set.

In addition, the results of this (and any other) study could be affected by the proportion of nonbankrupt firms to bankrupt firms in both the training and testing sets; that is, the population of all firms contains a certain proportion of firms on the verge of bankruptcy. This base rate may have an impact on a prediction technique's performance

in two ways. First, a technique may not work well when the firms of interest (i.e., those that are bankrupt) constitute a very small percentage of the population (i.e., a low base rate). This would be due to a technique's inability to identify the features necessary for classification. Second, there are differences in base rates between training samples and testing samples. If a classification model is built using a training sample with a certain base rate, does the model still work when the base rate in the test population is different? This issue is important for one more reason: If a classification model based on a certain base rate works across other proportions, it may be possible to build a model using a higher proportion of cases of interest than actually occur in the population.

To study the effects of this proportion on the predictive performance of the two techniques, we create three proportions (or base rates) for the testing set composition while holding the composition of the training set fixed at a 50/50 base rate. The first factor level (or base rate) can be a 50/50 proportion of bankrupt to nonbankrupt cases, the second level could be an 80/20 proportion (80 percent nonbankrupt, 20 percent bankrupt), and the third level could be an approximate 90/10 proportion. We do not really know the actual proportion of firms going bankrupt; the 80/20 and 90/10 cases should be close.

Within each of the three different testing set compositions, 20 different training–testing set pairs could be generated via Monte Carlo resampling from the original 129 firms. Thus, a total of 60 distinct training and testing data set pairs were generated from the original data. In each case, the training set and test set pairs contained unique firms (i.e., no overlap was allowed). This restriction provides a stronger test of a technique's performance. To summarize, neural networks and logistic regression models are developed using training sets of equal proportions of firms to determine the classification function but are evaluated with test sets containing 50/50, 80/20, and 90/ 10 base rates. (The data set used here is available from this book's Web site.)

Steps 3 through 6 relate to getting ready for a neural network experiment. We can use any neural network software package that implements the aforementioned backpropagation training algorithm to construct and test trained neural network models. We would have to decide on the size of the neural network, including the number of hidden layers and the number of neurons in the hidden layer. For example, one possible structure to use here is 5 input neurons (1 for each financial ratio), 10 hidden neurons, and 2 output neurons (1 indicating a bankrupt firm and the other indicating a nonbankrupt firm). (Figure 6.13 illustrates this network configuration.) Neural output values range from 0 to 1. Output node BR indicates a firm to be classified as likely to go bankrupt, and the node NBR, not so.

A user of a neural network has two difficult decisions to make in the training process (step 6): At what point has the neural network appropriately learned the relationships, and what is the threshold of error with regard to determining correct classifications of the test set? Typically, these issues are addressed by using training tolerances and testing tolerances that state the acceptable levels of variance for considering classifications as "correct."

Step 7 refers to the actual neural network training. In training the networks in this example, a heuristic backpropagation algorithm was used to ensure convergence (i.e., all firms in the training set classified correctly). The training set is presented to the neural network software repeatedly until the software has sufficiently learned the relationship between the attributes of the cases and whether the firm is distressed. Then, to accurately assess the prediction efficacy of the network, the holdout sample (i.e., test set) is presented to the network, and the number of correct classifications are noted (step 8).

In determining correct classifications, a testing threshold of 0.49 was used. Thus, the output node with a value over 0.5 was used to assess whether the network provided

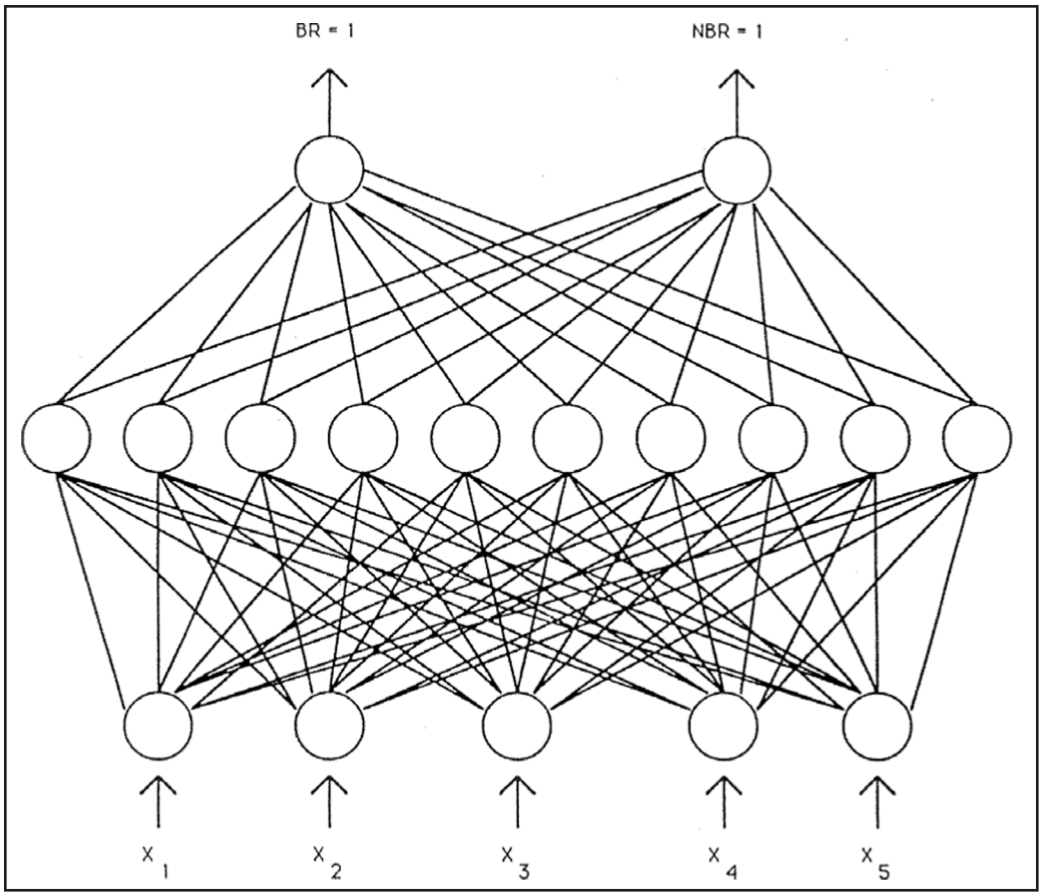


FIGURE 6.13 A Typical Neural Network Model for Bankruptcy Prediction

a correct classification. Cases in which both output neurons provided output levels either less than 0.5 or greater than 0.5 were automatically treated as misclassifications.

To compare the performance of the neural network against using classical statistical techniques, a logistic regression approach was implemented via SYSTAT, a statistical software package. Table 6.2 represents the average percentage of correct classifications provided by the two different techniques when evaluated by the 20 holdout samples for each of the three different test set base rates. When the testing sets contained an equal number of the two cases, neural networks correctly classified 97.5 percent of the holdout cases, whereas logistic regression was correct 93.25 percent of the time. Similarly, when the testing sets contained 20,070 bankrupt firms, neural networks classified at a 95.6 percent correct rate, whereas logistic regression correctly classified at a 92.2 percent rate.

A nonparametric test, the Wilcoxon test for paired observations, was undertaken to assess whether the correct classification percentages for the two techniques were significantly different. Those instances where statistically significant differences were found are indicated in Table 6.2 by footnotes. In general, neural networks performed significantly better than logistic regression.

Table 6.2 also illustrates the correct percentages of bankrupt firm predictions and nonbankrupt firm predictions. In the prediction of bankrupt cases, neural networks predicted significantly better than logistic regression for test sets of equal proportion,

TABLE 6.2 Performance Comparison of Neural Networks and Logistic Regression

	Test Proportions					
	50/50		***80/20***		***90/10***	
Criteria	***NN***	***LR***	***NN***	***LR***	***NN***	***LR***
Overall percentage of correct classification	97.5[a]	93.25	95.6[a]	92.2	95.68[b]	90.23
Bankrupt firm classification success rate	97.0[a]	91.90	92.0	92.0	92.5	95.0 ($p = .282$)
Nonbankrupt firm classification success rate	98.0[a]	95.5	96.5[a]	92.25	96.0[b]	89.75

[a] $p < .01$.
[b] $p < .05$.

at the same percentage when the ratio was 80/20, and a little worse (although not significantly) for 90/10 test sets. The neural networks clearly outperform the logistic regression model in the prediction of the nonbankrupt firms.

A number of studies in the recent past have investigated the performance of neural networks in predicting business failure. Typically, these studies have compared neural network performance to that of traditional statistical techniques such as discriminant analysis and logistic regression. In addition, some studies have compared neural networks to other artificial intelligence techniques, such as inductive learning methods (e.g., ID3). The purpose of this section is only to illustrate how a neural network project can be completed, not necessarily to argue that the neural networks do better in this problem domain.

Section 6.5 Review Questions

1. What parameters can be used to predict failure of a firm?
2. How were data divided between training and test sets for this experiment?
3. Explain what is meant by *resampling* in this context? How was resampling used for this problem?
4. What were the network parameters for this neural network experiment?
5. How was an output converted to mean bankrupt or nonbankrupt?
6. How did the neural network model compare with a logistic regression model in this experiment?

6.6 OTHER NEURAL NETWORK PARADIGMS

MLP-based neural networks described in this chapter thus far are just one specific type of neural networks. Literally hundreds of different neural networks have been proposed. Many are variants of the MLP model that you have already seen; they just differ in their implementations of input representation, learning process, output processing, and so on. But there are many types of neural networks that are quite different from the MLP model. Some of these are introduced later in this chapter. Others include radial basis function networks, probabilistic neural networks, generalized regression neural networks, and support vector machines. Many online resources describe details

of these types of neural networks. A good resource introduced in Chapter 4 is the e-book StatSoft (**statsoft.com/textbook/stathome.html**). The next subsection introduces some of the classic varieties of neural networks.

HOPFIELD NETWORKS

A neural network model of interest is the Hopfield network (Hopfield, 1982). John Hopfield showed in a series of papers in the 1980s how highly interconnected networks or nonlinear neurons can be extremely effective in computing. These networks provided a rapid computed solution for problems stated in terms of desired optima, often subject to constraints.

A general Hopfield network is a single large layer of neurons with total interconnectivity—that is, each neuron is connected to every other neuron. In addition, the output of each neuron may depend on its previous values. One use of Hopfield networks has been in solving constrained optimization problems, such as the classic traveling salesman problem (TSP). In this type of application, each neuron represents the desirability of a city n being visited in position m of a TSP tour. Interconnection weights are specified, representing the constraints of feasible solution to the TSP (e.g., forcing a city to appear in a tour only once). An energy function is specified, which represents the objective of the model solution process (e.g., minimize total distance in the TSP tour) and is used in determining when to stop the neural network evolution to a final state. The network starts with random neuron values and, using the stated interconnection weights, the neuron values are updated over time. Gradually, the neuron values stabilize, evolving into a final state (as driven by the global energy function) that represents a solution to the problem. At this point in the network evolution, the value of neuron (n,m) represents whether city n should be in location m of the TSP tour. While Hopfield and Tank (1985) and others claimed great success in solving the TSP, further research has shown those claims to be somewhat premature. Nonetheless, this novel approach to a classic problem offers promise for optimization problems, especially when technology allows for taking advantage of the inherent parallelism of neural networks.

Hopfield networks are distinct from feedforward networks because the neurons are highly interconnected, weights between neurons tend to be fixed, and there is no training per se. The complexity and challenge in using a Hopfield network for optimization problems is in the correct specification of the interconnection weights and the identification of the proper global energy function to drive the network evolution process.

SELF-ORGANIZING NETWORKS

Kohonen's network, also known as a self-organizing network is another neural network model. Such networks learn in an unsupervised mode. The biological basis of these models is the conjecture that some organization takes place in the human brain when an external stimulus is provided. Kohonen's algorithm forms "feature maps," where neighborhoods of neurons are constructed. These neighborhoods are organized such that topologically close neurons are sensitive to similar inputs into the model. Self-organizing maps, or self-organizing feature maps, can sometimes be used to develop some early insight into the data. For example, self-organizing maps could learn to identify clusters of data so that an analyst could build more refined models for each subset/cluster. In cases in which the analyst does not have a good idea of the number of classes or output or actual output class for any given pattern, the self-organizing maps can work well.

Section 6.6 Review Questions

1. List some of the different types of neural networks.
2. What is one key difference between an MLP network and a Kohonen network?
3. What is another name for a Kohonen network?
4. Briefly describe a Hopfield network.

6.7 APPLICATIONS OF ANN

ANN have been applied in many domains. A survey of their applications in finance can be found in Fadlalla and Lin (2001). There have been several tests of neural networks in financial markets. Collard (1990) stated that his neural network model for commodity training would have resulted in significant profits over other trading strategies. Kamijo and Tanigawa (1990) used a neural network to chart Tokyo Stock Exchange data. They found that the results of the model would beat a "buy and hold" strategy. Finally, a neural model for predicting percentage change in the S&P 500 five days ahead, using a variety of economic indicators, was developed (Fishman et al., 1991). The authors claim that the model has provided more accurate prediction than alleged experts in the field using the same indicators.

Neural networks have been successfully trained to determine whether loan applications should be approved (Gallant, 1988). It has also been shown that neural networks can predict mortgage applicant solvency better than mortgage writers (Collins et al., 1988). Predicting rating of corporate bonds and attempting to predict their profitability is another area where neural networks have been successfully applied (see Dutta and Shakhar, 1988; and Surkan and Singleton, 1990). Neural networks outperformed regression analysis and other mathematical modeling tools in predicting bond rating and profitability. The main conclusion reached was that neural networks provided a more general framework for connecting financial information of a firm to the respective bond rating.

Fraud prevention is another area of neural network application in business. Chase Manhattan Bank successfully used neural networks in dealing with credit card fraud (Rochester, 1990), with the neural network models outperforming traditional regression approaches. Also, neural networks have been used in the validation of bank signatures (see Francett, 1989; and Mighell, 1989). These networks identified forgeries significantly better than any human expert.

Another significant area of statistical application of neural networks is in time-series forecasting. Several studies have attempted to use neural networks for time-series prediction. Examples include Fozzard et al. (1989), Tang et al. (1991), and Hill et al. (1994). The general conclusion is that neural networks appear to do at least as well as the Box-Jenkins forecasting technique.

Because neural networks have been a subject of intense study since late 1980s, there have been many applications of as well as experiments with applications. You can do simple Web searches to find recent examples in addition to the ones listed in this chapter. Other recent reports include live intrusion tracking (see Thaler, 2002), Web content filtering (Lee et al., 2002), exchange rate prediction (Davis et al., 2001), and hospital bed allocation (Walczak et al., 2002). Newer applications are emerging in health care and medicine. See Application Case 6.5, for example.

Application Case 6.5

Neural Networks for Breast Cancer Diagnosis

ANN have proven to be a useful tool in pattern recognition and classification tasks in diverse areas, including clinical medicine. Despite the wide applicability of ANN, the large amount of data required for training makes using them an unsuitable classification technique when the available data are scarce. Magnetic resonance spectroscopy (MRS) plays a pivotal role in the investigation of cell biochemistry and provides a reliable method for detection of metabolic changes in breast tissue. The scarcity of data and the complexity of interpretation of relevant physiological information impose extra demands that prohibit the applicability of most statistical and machine learning techniques developed. Knowledge-based artificial neural networks (KBANN) help to prevail over such difficulties and complexities. A KBANN combines knowledge from a domain, in the form of simple rules, with connectionist learning. This combination trains the network through the use of small sets of data (as is typical of medical diagnosis tasks). The primary structure is based on the dependencies of a set of known domain rules, and it is necessary to refine those rules through training.

The KBANN process consists of two algorithms. One is the Rules-to-Network algorithm, in which the main task is the translation process between a knowledge base containing information about a domain theory and the initial structure of a neural network. This algorithm maps the structure of an approximately correct domain theory, with all the rules and their dependencies, into a neural network structure. The defined network is then trained using the backpropagation learning algorithm.

Feedback mechanisms, which inhibit or stimulate the growth of normal cells, control the division and replacement of cells in normal tissues. In the case of tumors, that process is incapable of controlling the production of new cells, and the division is done without any regard to the need for replacement, disturbing the structure of normal tissue. Changes observed in phospholipid metabolite concentrations, which are associated with differences in cell proliferation in malignant tissues, have served as the basic inputs for the identification of relevant features present in malignant or cancerous tissues but not in normal tissues. The abnormal levels of certain phospholipid characteristics are considered indicators of tumors. These include several parameters, such as PDE, PME, Pi, PCr, γATP, αATP, and βATP. KBANN produced an accurate tumor classification of 87 percent from a set of 26, with an average pattern error of 0.0500 and a standard deviation of 0.0179.

Sources: M. Sordo, H. Buxton, and D. Watson, "A Hybrid Approach to Breast Cancer Diagnosis," in *Practical Applications of Computational Intelligence Techniques*, Vol. 16, in L. Jain and P. DeWilde (eds.), Kluwer, Norwell, MA, 2001, **acl.icnet.uk/PUBLICATIONS/sordo/chapter2001.pdf** (accessed March 2006).

In general, ANN are suitable for problems whose inputs are both categorical and numeric, and where the relationships between inputs and outputs are not linear or the input data are not normally distributed. In such cases, classical statistical methods may not be reliable enough. Because ANN do not make any assumptions about the data distribution, their power is less affected than traditional statistical methods when data are not properly distributed. Finally, there are cases in which the neural networks simply provide one more way of building a predictive model for the situation at hand. Given the ease of experimentation using the available software tools, it is certainly worth exploring the power of neural networks in any data modeling situation.

Section 6.7 Review Questions

1. List some applications of neural networks in accounting/finance.
2. What are some engineering applications of neural networks?
3. How have neural networks been used in the health care field?
4. What are some applications of neural networks in information security?
5. Conduct a Web search to identify homeland security applications of neural networks.

6.8 A NEURAL NETWORK SOFTWARE DEMONSTRATION

Online Tutorial T4 provides a software demonstration of using neural networks. That section is used, with permission, from STATISTICA Software Tutorial. Students and professors using this book are eligible to receive a six-month license to use STATISTICA software for completion of the exercises in Chapters 4 and 6. Request for this copy of the software is to be made by the instructor by completing the coupon available on the Companion Website www.prenhall.com/turban. Note that similar software projects can also be completed by using tools identified in Technology Insights 6.4.

Chapter Highlights

- Neural computing involves a set of methods that emulate the way the human brain works. The basic processing unit is a neuron. Multiple neurons are grouped into layers and linked together.
- In a neural network, the knowledge is stored in the weight associated with each connection between two neurons.
- Backpropagation is the most popular paradigm in business applications of neural networks. Most business applications are handled using this algorithm.
- A backpropagation-based neural network consists of an input layer, an output layer, and a certain number of hidden layers (usually one). The nodes in one layer are fully connected to the nodes in the next layer. Learning is done through a trial-and-error process of adjusting the connection weights.
- Each node at the input layer typically represents a single attribute that may affect the prediction.
- Neural network learning can occur in supervised or unsupervised mode.
- In supervised learning mode, the training patterns include a correct answer/classification/forecast.
- In unsupervised learning mode, there are no known answers. Thus, unsupervised learning is used for clustering or exploratory data analysis.
- The usual process of learning in a neural network involves three steps: (1) compute temporary outputs based on inputs and random weights, (2) compute outputs with desired targets, and (3) adjust the weights and repeat the process.
- The delta rule is commonly used to adjust the weights. It includes a learning rate and a momentum parameter.
- Developing neural network–based systems requires a step-by-step process. It includes data preparation and preprocessing, training and testing, and conversion of the trained model into a production system.
- Neural network software is available to allow easy experimentation with many models. Neural network modules are included in all major data mining software tools. Specific neural network packages are also available. Some neural network tools are available as spreadsheet add-ins.
- After a trained network has been created, it is usually implemented in end-user systems through programming languages such as C++, Java, and Visual Basic. Most neural network tools can generate codes for the trained network in these languages.
- Many neural network models beyond backpropagation exist, including radial basis functions, support vector machines, Hopfield networks, and Kohonen's self-organizing maps.
- Neural network applications abound in almost all business disciplines as well as in virtually all other functional areas.
- Business applications of neural networks included finance, firm failure prediction, time series forecasting, and so on.
- New applications of neural networks are emerging in health care, security, and so on.

Key Terms

- adaptive resonance theory (ART)
- artificial neural network (ANN)
- axon
- backpropagation
- black-box testing
- connection weight
- dendrite
- hidden layer
- Kohonen self-organizing feature map
- learning algorithm
- learning rate

- momentum
- neural computing
- neural network
- neuron
- nucleus
- parallel processing
- pattern recognition
- perceptron
- processing element (PE)
- self-organizing
- sigmoid (logical activation) function
- summation function
- supervised learning
- synapse
- threshold value
- topology
- transformation (transfer) function
- unsupervised learning

QUESTIONS FOR DISCUSSION

1. Compare artificial and biological neural networks. What aspects of biological networks are not mimicked by artificial ones? What aspects are similar?
2. The performance of ANN relies heavily on the summation and transformation functions. Explain the combined effects of the summation and transformation functions and how they differ from statistical regression analysis.
3. ANN can be used for both supervised and unsupervised learning. Explain how they learn in a supervised mode and in an unsupervised mode.
4. Explain the difference between a training set and a testing set. Why do we need to differentiate them? Can the same set be used for both purposes? Why or why not?
5. Say that a neural network has been constructed to predict the creditworthiness of applicants. There are two output nodes: one for yes (1 = yes, 0 = no) and one for no (1 = no, 0 = yes). An applicant receives a score of 0.83 for the "yes" output node and a 0.44 for the "no" output node. Discuss what may have happened and whether the applicant is a good credit risk.
6. Everyone would like to make a great deal of money on the stock market. Only a few are very successful. Why is using an ANN a promising approach? What can it do that other decision support technologies cannot do? How could it fail?

Exercises

Teradata University and Other Hands-on Exercises

1. Go to Teradata Student Network Web site (at **teradatastudentnetwork.com**) or the URL given by your instructor. Locate Web seminars related to data mining and neural networks. Specifically, view the seminar given by Professor Hugh Watson at the SPIRIT2005 conference at Oklahoma State University. Then answer the following questions:
   a. Which real-time application at Continental Airlines may have used a neural network?
   b. What inputs and outputs can be used in building a neural network application?
   c. Given that Continental's data mining applications are real-time, how might Continental implement a neural network in practice?
   d. What other neural network applications would you propose for the airline industry?

2. Go to Teradata Student Network Web site (at **teradatastudentnetwork.com**) or the URL given by your instructor. Locate the Harrah's case. Read the case and answer the following questions:
   a. Which of the Harrah's data applications are most likely implemented using neural networks?
   b. What other applications could Harrah's develop using the data it is collecting from its customers?
   c. What are some concerns you might have as a customer at this casino?

3. This exercise relates to the sample project in this chapter. Bankruptcy prediction problem can be viewed as a problem of classification. The data set you will be using for this problem includes five ratios that have been computed from the financial statements of real-world firms. These five ratios have been used in studies involving bankruptcy prediction. The first sample includes data on firms that went bankrupt and firms that didn't. This will be your training sample for the neural network. The second sample of 10 firms also consists of some bankrupt firms and some nonbankrupt firms. Your goal is to train a neural network, using the first 20 data, and then test its performance on the other

10 data. (Try to analyze the new cases yourself manually before you run the neural network and see how well you do.) The following tables show the training sample and test data you should use for this exercise:

Training Sample

Firm	WC/TA	RE/TA	EBIT/TA	MVE/TD	S/TA	BR/NB
1	0.165	0.1192	0.2035	0.813	1.6702	1
2	0.1415	0.3868	0.0681	0.5755	1.0579	1
3	0.5804	0.3331	0.081	1.1964	1.3572	1
4	0.2304	0.296	0.1225	0.4102	3.0809	1
5	0.3684	0.3913	0.0524	0.1658	1.1533	1
6	0.1527	0.3344	0.0783	0.7736	1.5046	1
7	0.1126	0.3071	0.0839	1.3429	1.5736	1
8	0.0141	0.2366	0.0905	0.5863	1.4651	1
9	0.222	0.1797	0.1526	0.3459	1.7237	1
10	0.2776	0.2567	0.1642	0.2968	1.8904	1
11	0.2689	0.1729	0.0287	0.1224	0.9277	0
12	0.2039	–0.0476	0.1263	0.8965	1.0457	0
13	0.5056	–0.1951	0.2026	0.538	1.9514	0
14	0.1759	0.1343	0.0946	0.1955	1.9218	0
15	0.3579	0.1515	0.0812	0.1991	1.4582	0
16	0.2845	0.2038	0.0171	0.3357	1.3258	0
17	0.1209	0.2823	–0.0113	0.3157	2.3219	0
18	0.1254	0.1956	0.0079	0.2073	1.489	0
19	0.1777	0.0891	0.0695	0.1924	1.6871	0
20	0.2409	0.166	0.0746	0.2516	1.8524	0

Test Data

Firm	WC/TA	RE/TA	EBIT/TA	MVE/TD	S/TA	BR/NB
A	0.1759	0.1343	0.0946	0.1955	1.9218	?
B	0.3732	0.3483	–0.0013	0.3483	1.8223	?
C	0.1725	0.3238	0.104	0.8847	0.5576	?
D	0.163	0.3555	0.011	0.373	2.8307	?
E	0.1904	0.2011	0.1329	0.558	1.6623	?
F	0.1123	0.2288	0.01	0.1884	2.7186	?
G	0.0732	0.3526	0.0587	0.2349	1.7432	?
H	0.2653	0.2683	0.0235	0.5118	1.835	?
I	0.107	0.0787	0.0433	0.1083	1.2051	?
J	0.2921	0.239	0.0673	0.3402	0.9277	?

Describe the results of the neural network prediction, including software, architecture, and training information. Submit the trained network file(s) so that your instructor can load and test your network.

4. For this exercise, your goal is to build a model to identify inputs or predictors that differentiate risky customers from others (based on patterns pertaining to previous customers) and then use those inputs to predict the

new risky customers. This sample case is typical for this domain.

The sample data to be used in this exercise is posted on www.prenhall.com/turban, file name: CreditRisk.xls. The data set has 425 cases and 15 variables pertaining to past and current customers who borrowed from a bank for various reasons. The data set contains various information related to the customers, financial standing, reason to loan, employment, demographic information, and so on, and finally the outcome or dependent variable for credit standing, classifying each case as good or bad, based on the institution's past experience.

You should take 400 of the cases as training cases and use the other 25 for testing. Then build a neural network model to learn the characteristics of the problem and test its performance on the other 25 cases. Report on your model's learning and testing performance. Prepare a report that identifies the neural network architecture, training parameters, and resulting performance on the test set.

(This exercise is courtesy of StatSoft, Inc., based on a German data set from **ftp://ftp.ics.uci.edu/pub/machine-learning-databases/statlog/german** renamed CreditRisk and altered.)

5. Forecasting box-office receipts for a particular motion picture is an interesting challenge. Despite the difficulty associated with the unpredictable nature of the problem domain, several researchers have tried to predict the total box-office receipt of motion pictures after a movie's initial theatrical release. In this problem, you explore forecasting the financial performance of a movie at the box office before its theatrical release by converting the forecasting problem into a classification problem. That is, rather than forecasting the point estimate of box-office receipts, you classify a movie based on its box-office receipts in one of nine categories, ranging from flop to blockbuster, taking into account a number of factors decided by feedback received from the industry experts and previous studies. The following is the list of the variables:

Attributes	Values- range	Type
MPAA rating	5 possible rating categories: G, PG, PG-13, R, NR	Binary (0,1)
Competition	3 pseudo-variables: high, medium, low competition	Binary (0,1)
Star value	3 variables of degree of star value: A+/A (high), B (medium), C (insignificant)	Binary (0,1)
Content category (genre)	10 categories: sci-fi, historic epic drama, modern drama, politically related, thriller, horror, comedy, cartoon, action, documentary	Binary (0,1)
Technical effects	3 binary independent variables: high, medium, low technical effects ratings	Binary (0,1)
Sequel	1 variable to specify whether a movie is a sequel	Binary (0,1)
Number of screens	Continuous variable	Positive integer

Each categorical-independent variable (except the genre variable) is converted into a 1-of-*N* binary representation. For example, the 5 MPAA ratings are represented as five 0–1 variables. In the process of value assignment, all such pseudo-representations of a categorical variable are given the value of 0, except the one that holds true for the current case, which is given the value of 1. For a movie of rating PG, the second input variable is at level 1, the others (1 and 3–5) are at level 0.

The variable of interest here is box-office gross revenues. A movie based on its box-office receipts is classified in one of nine categories, ranging from a flop to a blockbuster. The dependent variable can be converted into nine classes, using the following breakpoints:

Class Number	Range (in millions)
1	<1 (flop)
2	>1 and <10
3	>10 and <20
4	>20 and <40
5	>40 and <65
6	>65 and <100
7	>100 and <150
8	>150 and <200
9	>200 (blockbuster)

Download the training set data from www.prenhall.com/turban, file name: movietrain.xls, which has 184 records and is in Microsoft Excel format. Use the data description here to understand the domain and the problem you are trying to solve. Pick and choose your independent variables; develop at least three classification models (e.g., decision tree, logistic regression, neural networks). Compare the accuracy results (using 10-fold cross validation and percentage split techniques), use confusion matrices, and comment on the outcome. Test the models developed on the test set

(see www.prenhall.com/turban, file name: movietest.xls, 29 records) and analyze the results with different models and come up with the best classification model, supporting it with your results.

Team Assignments and Role-Playing

1. Consider the following set of data that relates daily electricity usage as a function of outside high temperature (for the day):

Temperature, X	Kilowatts, Y
46.8	12,530
52.1	10,800
55.1	10,180
59.2	9,730
61.9	9,750
66.2	10,230
69.9	11,160
76.8	13,910
79.7	15,110
79.3	15,690
80.2	17,020
83.3	17,880

   a. Plot the raw data. What pattern do you see? What do you think is really affecting electricity usage?
   b. Solve this problem with linear regression $Y = a + bX$ (in a spreadsheet). How well does this work? Plot your results. What is wrong? Calculate the sum-of-the-squares error and R^2.
   c. Solve this problem by using nonlinear regression. We recommend a quadratic function, $Y = a + b_1X + b_2X^2$. How well does this work? Plot your results. Is anything wrong? Calculate the sum-of-the-squares error and R^2.
   d. Break up the problem into three sections (look at the plot) and solve it using three linear regression models—one for each section. How well does this work? Plot your results. Calculate the sum-of-the-squares error and R^2. Is this modeling approach appropriate? Why or why not?
   e. Build a neural network to solve the original problem. (You may have to scale the X and Y values to be between 0 and 1.) Train it (on the entire set of data) and solve the problem (i.e., make predictions for each of the original data items). How well does this work? Plot your results. Calculate the sum-of-the-squares error and R^2.
   f. Which method works best and why?

2. Build a real-world neural network. Using demo software downloaded from the Web (e.g., Braincel, at **promland.com**, or another site), identify real-world data (e.g., start searching on the Web at **ics.uci.edu/~mlearn/MLRepository.html** or use data from an organization with which someone in your group has a contact) and build a neural network to make predictions. Topics might include sales forecasts, predicting success in an academic program (e.g., predict GPA from high school rating and SAT scores; being careful to look out for "bad" data, such as GPAs of 0.0), or housing prices; or survey the class for weight, gender, and height and try to predict height based on the other two factors. (Hint: Use U.S. census data, on this book's Web site or at **census.gov**, by state, to identify a relationship between education level and income.) How good are your predictions? Compare the results to predictions generated using standard statistical methods (regression). Which method is better? How could your system be embedded in a DSS for real decision making?

3. For each of the following applications, would it be better to use a neural network or an expert system? Explain your answers, including possible exceptions or special conditions.
   a. Diagnosis of a well-established but complex disease
   b. **Price-lookup** subsystem for a high-volume merchandise seller
   c. Automated voice-inquiry processing system
   d. Training of new employees
   e. Handwriting recognition

4. Consider the following data set, which includes three attributes and a classification for admission decisions into an MBA program:

GMAT	GPA	Quant. GMAT Percentile	Decision
650	2.75	35	NO
580	3.50	70	NO
600	3.50	75	YES
450	2.95	80	NO
700	3.25	90	YES
590	3.50	80	YES
400	3.85	45	NO
640	3.50	75	YES
540	3.00	60	?
690	2.85	80	?
490	4.00	65	?

   a. Using the data given here as examples, develop your own manual expert rules for decision making.
   b. Build a decision tree using SPRINT (Gini index). You can build it by using manual calculations or use a spreadsheet to perform the basic calculations.

c. Build another decision tree, using the entropy and information gain (ID3) approach. You can use a spreadsheet calculator for this exercise.

d. Although the data set here is extremely small, try to build a little neural network for it.

e. Use automated decision tree software (e.g., See5; download a trial version from **rulequest.com**) to build a tree for these data.

f. Report the predictions on the last three observations from each of the five classification approaches.

g. Comment on the similarity and differences of the approaches. What did you learn from this exercise?

5. You have worked on neural networks and other data mining techniques. Give examples of where each of these has been used. Based on your knowledge, how would you differentiate among these techniques? Assume that a few years from now, you come across a situation in which neural network or other data mining techniques could be used to build an interesting application for your organization. You have an intern working with you to do the grunt work. How will you decide whether the application is appropriate for a neural network or for another data mining model? Based on your homework assignments, what specific software guidance can you provide to get your intern to be productive for you quickly? Your answer for this question might mention the specific software, describe how to go about setting up the model/neural network, and validate the application.

Internet Exercises

1. Explore the Web sites of several neural network vendors, such as California Scientific Software (**calsci.com**), NeuralWare (**neuralware.com**), and Ward Systems Group (**wardsystems.com**), and review some of their products. Download at least two demos and install, run, and compare them.

2. There is a very good repository of data that has been used to test the performance of neural network and other machine learning algorithms. This repository can be accessed at **ics.uci.edu/~mlearn/MLRepository.html**. Some of the data sets are really meant to test the limits of current machine learning algorithms and compare their performance against new approaches to learning. However, some of the smaller data sets can be useful for exploring the functionality of the software you might download in Internet Exercise 1 or the software that is available as companion software with this book, such as STATISTICA Data Miner. Download at least one data set from the UCI repository (e.g., Credit Screening Databases, Housing Database). Then apply neural networks as well as decision tree methods, as appropriate. Prepare a report on your results. (Some of these exercises could also be completed in a group or may even be proposed as semester-long projects for term papers and so on.)

3. Go to **calsci.com** and read about various business applications. Prepare a report that summarizes the applications.

4. Go to **nd.com**. Read about the company's applications in investment and trading. Prepare a report about them.

5. Go to **nd.com**. Download the trial version of Neurosolutions for Excel and experiment with it, using one of the data sets from the exercises in this chapter. Prepare a report about your experience with the tool.

6. Go to **neoxi.com**. Identify at least two software tools that have not been mentioned in this chapter. Visit Web sites of those tools and prepare a brief report on the capabilities of those tools.

7. Go to **neuroshell.com**. Look at Gee Whiz examples. Comment on the feasibility of achieving the results claimed by the developers of this neural network model.

8. Go to **easynn.com**. Download the trial version of the software. After the installation of the software, find the sample file called Houseprices.tvq. Retrain the neural network and test the model by supplying some data. Prepare a report about your experience with this software.

9. Visit **statsoft.com**. Go to Downloads and download at least three white papers of applications. Which of these applications may have used neural networks?

10. Go to **neuralware.com**. Prepare a report about the products the company offers.

End of Chapter Case

Sovereign Credit Ratings Using Neural Networks

Companies such as Standard & Poor's Corporation, Moody's Investors Service, and Fitch Ratings provide alphabetical indicators of credit risk. Over a long period of time, these ratings have been in use to assess companies and financial institutions. However, issuing sovereign ratings is relatively new but has seen rapid expansion in recent years. The number of rated sovereigns grew from 17 in 1989 to 63 in 1998. Sovereign credit ratings are receiving a lot of global importance, as both a measure of credit risk for a country and of the firms that operate within the country. The Bank for International Settlements (**bis.org**) has been at the forefront in using credit ratings prominently in determining capital adequacy.

Multiple factors are used in performing credit risk analysis for sovereign country ratings. These include financial ratios; the economic, political, and regulatory environment; and industry trends. In the context of quantitative models, using financial, economic, and business data to arrive at a credit rating is a challenging process due to the complex and nonlinear interactions between different variables. However, this risk assessment process lacks a well-defined theory, which makes it difficult to apply conventional mathematical or rule-based techniques, although there are numerous quantitative approaches.

ANN are suited for modeling the determinants of ratings because they do not require prior specification of theoretical models. Their particular strength in classifying outcomes lends itself to producing a calibrated rating scale. ANN provide an alternative to the econometric approaches in that there are no assumptions with respect to the underlying properties and relationships within the data. ANN score above all other models in deriving meaning from complicated or imprecise data. A successful ANN implementation will generate a system of relationships that has been learned from observing past examples, and it can generalize and apply these lessons to new examples.

Bennell et al. (2006) compared ANN implementations to the standard credit risk analysis approach of probit. The sample set included 1,383 annual (end-of-calendar-year) observations of long-term foreign-currency sovereign credit ratings, assigned by 11 international credit rating agencies to 70 sovereign borrowers during the period from 1989 to 1999. The input variables were chosen to be consistent with the factors stressed in both the theoretical and empirical literature as determining the capacity and willingness of sovereign borrowers to service external debt. Some of the economic indicators that were chosen as explanatory variables are as follows:

Input Variable	Description
External debt/export	Total external debt relative to exports for the previous year
Fiscal balance	Average annual central government deficit (−) or surplus (+) relative to GDP for the previous three years (percentage)
External balance	Average annual current account balance relative to GDP for the previous three years (percentage)
Rate of inflation	Average annual consumer price inflation rate for the previous three years (percentage)
GDP per capita	GDP for the previous year (U.S. dollars)
GDP growth	Average annual real GDP growth on a year-over-year basis for the previous four years (percentage)
Development indicator	International Monetary Fund country classification for the current year (1 = industrial, 0 = not industrial)

Source: Adapted from J. Bennell, D. Crabbe, S. Thomas, and O. Gwilym, "Modelling Sovereign Credit Ratings: Neural Networks Versus Ordered Probit," *Expert Systems with Applications*, April 2006, pp. 415–425.

In addition to the macroeconomic variables specified here, two sets of indicator variables were included to capture effects on sovereign ratings in a given year: sovereign rating assigned by other rating agencies and the sovereign's location in a specific geographic region.

The data were split into three distinct groups: training (in sample), testing (out of sample), and cross-validation. A target split of the data of 65 percent, 20 percent, and 15 percent was set for training, testing, and cross-validation, respectively. However, the partitioning of the data was constrained by the 16 replications of ratings by different agencies. Multilayer feedforward networks, each with one hidden layer, were implemented. The number of neurons in the hidden layer was optimized by sequentially adding additional neurons until no further improvement in out-of-sample classification was achieved.

The authors used different learning rates and momentum values, ranging from 0.7 to 1. The training was performed for different number of cycles (called *epochs*): 1,000, 2,000, 3,000, 4,000, and 5,000. By comparing the mean absolute error across different trials, the authors selected a generalized feedforward (GFF) network as the best-performing network.

Multiple criteria are important in assessing the performance of neural networks. It is important to distinguish between within-sample modeling accuracy and out-of-sample predictive accuracy. Further informative criteria are included as well: percentage correctly classified within two and three rating notches, maximum deviation from correct rating, and mean absolute error. The neural network models were tested a number of times, and the authors reported the average performances as well as the best performance on each performance criterion.

The rating agencies collectively assign foreign currency sovereign ratings by assessing factors consistent with those stressed by the theory as vital to determine sovereign capacity and willingness to service external debt. In the case of the classification and regression models, the best model was obtained from training for 5,000 epochs. Keeping in mind the percentage of ratings accurately classified, the classification-based neural network model performs its best at 42.4 percent correct, with an average performance of 40.4 percent, followed by the regression-based neural network model, with 33.9 percent and 34.6 percent for best and average performance, respectively. Correctly classified ratings within one notch were achieved in 67.3 percent and 73.5 percent of cases and an average performance of 63.6 percent and 68.9 percent cases for the classification and regression models, respectively. Within three notches, the regression-based neural network model accurately classifies on average 96.7 percent of ratings, with the other two models approaching the 90 percent accuracy mark.

In comparing the two neural network models, the regression model achieves a lower percentage of correctly classified ratings than the classification model, but it tends to deviate much less if a rating is not precisely correct.

The findings indicate that ANN for fitting credit ratings for corporations as practiced by the major rating agencies (e.g., Moody's, S&P) can be successfully applied to sovereign ratings. An analyst's role and partly subjective process of assigning a credit rating cannot be eliminated by neural networks. However, it appears that ANN could inform and support the analyst in the decision-making process.

Sources: J. Bennell, D. Crabbe, S. Thomas, and O. Gwilym, "Modelling Sovereign Credit Ratings: Neural Networks versus Ordered Probit," *Expert Systems with Applications*, April 2006, pp. 415–425; and S. Hoti and M. McAleer, *Country Risk Ratings: An International Comparison*, **e.u-tokyo.ac.jp/cirje/research/papers/mcaleer/mcaleer4.pdf** (accessed March 2006).

QUESTIONS FOR THE CASE

1. What are sovereign ratings? Why are sovereign ratings important?
2. What role do rating agencies play?
3. What is the role of neural networks in sovereign ratings? Do you think we should entirely rely on neural network prediction?
4. What would you conclude from the outcomes of neural network–based rating prediction experiments?
5. You are a credit analyst at Standard & Poor's. You have been asked to rate India's sovereign credit. What factors would you consider in arriving at a credit rating, and how would you use neural networks in arriving at a result? Explain.

References

Ainscough, T.L., and J.E. Aronson. (1999). "A Neural Networks Approach for the Analysis of Scanner Data." *Journal of Retailing and Consumer Services*, Vol. 6.

Altman, E.I. (1968). "Financial Ratios, Discriminant Analysis and the Prediction of Corporate Bankruptcy." *Journal of Finance*, Vol. 23.

Bennell, J., D. Crabbe, S. Thomas, and O. Gwilym. (2006, April). "Modelling Sovereign Credit Ratings: Neural Networks versus Ordered Probit," *Expert Systems with Applications*.

Collard, J.E. (1990). "Commodity Trading with a Neural Net." *Neural Network News*, Vol. 2, No. 10.

Collins, E., S. Ghosh, C.L. and Scofield. (1988). "An Application of a Multiple Neural Network Learning System to Emulation of Mortgage Underwriting Judgments," *IEEE International Conference on Neural Networks*.

Davis, J.T., A. Episcopos, and S. Wettimuny. (2001). "Predicting Direction Shifts on Canadian–U.S. Exchange Rates with Artificial Neural Networks," *International Journal of Intelligent Systems in Accounting, Finance and Management*, Vol. 10, No. 2.

Dutta, S., and S. Shakhar. (1988, July 24–27). "Bond-Rating: A Non-Conservative Application of Neural

Networks" *Proceedings of the IEEE International Conference on Neural Networks*, San Diego.

Fadlalla, A., and C. Lin. (2001). "An Analysis of the Applications of Neural Networks in Finance." *Interfaces*, Vol. 31, No. 4.

Fishman, M., D. Barr, and W. Loick. (1991, April). "Using Neural Networks in Market Analysis," *Technical Analysis of Stocks and Commodities*.

Fozzard, R., G. Bradshaw, and L. Ceci. (1989). "A Connectionist Expert System for Solar Flare Forecasting," in D.S. Touretsky (ed.), *Advances in Neural Information Processing Systems* Vol.1. San Mateo, CA: Kaufman Publishing.

Francett, B. (1989, January). "Neural Nets Arrive." *Computer Decisions*.

Gallant, S. (1988, February). "Connectionist Expert Systems," *Communications of the ACM*, Vol. 31, No. 2.

Haykin, S.S. (1999). *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, NJ: Prentice Hall.

Hill, T., T. Marquez, M. O'Connor, and M. Remus. (1994). "Neural Network Models for Forecasting and Decision Making," *International Journal of Forecasting*, Vol. 10.

Hopfield, J. (1982, April). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities." *Proceedings of National Academy of Science*, Vol. 79, No. 8.

Hopfield, J.J., and D.W. Tank. (1985). "Neural Computation of Decisions in Optimization Problems," *Biological Cybernetics*, Vol. 52.

Kamijo, K., and T. Tanigawa. (1990, June 7–11). "Stock Price Pattern Recognition: A Recurrent Neural Network Approach," *International Joint Conference on Neural Networks*, San Diego.

Lee, P.Y., S.C. Hui, and A.C.M. Fong. (2002, September/October). "Neural Networks for Web Content Filtering." *IEEE Intelligent Systems*.

Liang, T.P. (1992). "A Composite Approach to Automated Knowledge Acquisition." *Management Science*, Vol. 38, No. 1.

McCulloch, W.S., and W.H. Pitts. (1943). "A Logical Calculus of the Ideas Imminent in Nervous Activity." *Bulletin of Mathematical Biophysics*, Vol. 5.

Mighell, D. (1989). "Back-Propagation and Its Application to Handwritten Signature Verification," in D. S. Touretsky (ed.), *Advances in Neural Information Processing Systems*. San Mateo, CA: Kaufman.

Minsky, M., and S. Papert. (1969). *Perceptrons*. Cambridge, MA: MIT Press.

Principe, J.C., N.R. Euliano, and W.C. Lefebvre. (2000). *Neural and Adaptive Systems: Fundamentals Through Simulations*. New York: Wiley.

Rochester, J. (ed.). (1990, February). "New Business Uses for Neurocomputing." *I/S Analyzer*.

Surkan, A., and J. Singleton. (1990). "Neural Networks for Bond Rating Improved by Multiple Hidden Layers." *Proceedings of the IEEE International Conference on Neural Networks*, Vol. 2.

Tang, Z., C. de Almieda, and P. Fishwick. (1991). "Time-Series Forecasting Using Neural Networks vs. Box-Jenkins Methodology." *Simulation*, Vol. 57, No. 5.

Thaler, S.L. (2002, January/February). "AI for Network Protection: LITMUS:—Live Intrusion Tracking via Multiple Unsupervised STANNOs." *PC AI*.

Walczak, S., W.E. Pofahi, and R.J. Scorpio. (2002). "A Decision Support Tool for Allocating Hospital Bed Resources and Determining Required Acuity of Care." *Decision Support Sy*stems, Vol. 34, No. 4.

Wilson, R., and R. Sharda. (1994). "Bankruptcy Prediction Using Neural Networks." *Decision Support Systems*, Vol. 11.

Zahedi, F. (1993). *Intelligent Systems for Business: Expert Systems with Neural Networks*. Belmont, CA: Wadsworth.